EXHIBIT 1

- **Tor Overview**
- Users of Tor
- Tor People
- Jobs
- Sponsors
- Financial Reports
- Projects
- Documentation

Tor Tip

Tor is written for and supported by people like you. Donate today!

Tor: Overview

Topics

- Overview
- Why we need Tor
- The Solution
- Staying anonymous
- The future of Tor

Overview

The Tor network is a group of volunteer-operated servers that allows people to improve their privacy and security on the Internet. Tor's users employ this network by connecting through a series of virtual tunnels rather than making a direct connection, thus allowing both organizations and individuals to share information over public networks without compromising their privacy. Along the same line, Tor is an effective censorship circumvention tool, allowing its users to reach otherwise blocked destinations or content. Tor can also be used as a building block for software developers to create new communication tools with built-in privacy features.

Individuals use Tor to keep websites from tracking them and their family members, or to connect to news sites, instant messaging services, or the like when these are blocked by their local Internet providers. Tor's hidden services let users publish web sites and other services without needing to reveal the location of the site. Individuals also use Tor for socially sensitive communication: chat rooms and web forums for rape and abuse survivors, or people with illnesses.

Journalists use Tor to communicate more safely with whistleblowers and dissidents. Non-governmental organizations (NGOs) use Tor to allow their workers to connect to their home website while they're in a foreign country, without notifying everybody nearby that they're working with that organization.

Groups such as Indymedia recommend Tor for safeguarding their members' online privacy and security. Activist groups like the Electronic Frontier Foundation (EFF) recommend Tor as a mechanism for maintaining civil liberties online. Corporations use Tor as a safe way to conduct competitive analysis, and to protect sensitive procurement patterns from eavesdroppers. They also use it to replace traditional VPNs, which reveal the exact amount and timing of communication. Which locations have employees working late? Which locations have employees consulting job-hunting websites? Which research divisions are communicating with the company's patent lawyers?

A branch of the U.S. Navy uses Tor for open source intelligence gathering, and one of its teams used Tor while deployed in the Middle East recently. Law enforcement uses Tor for visiting or surveilling web sites without leaving government IP addresses in their web logs, and for security during sting operations.

The variety of people who use Tor is actually part of what makes it so secure. Tor hides you among the other users on the network, so the more populous and diverse the user base for Tor is, the more your anonymity will be protected.

Why we need Tor

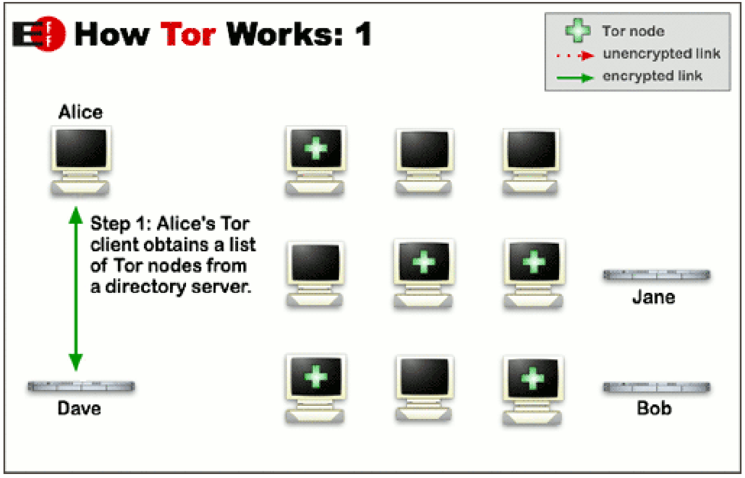
Using Tor protects you against a common form of Internet surveillance known as "traffic analysis." Traffic analysis can be used to infer who is talking to whom over a public network. Knowing the source and destination of your Internet traffic allows others to track your behavior and interests. This can impact your checkbook if, for example, an e-commerce site uses price discrimination based on your country or institution of origin. It can even threaten your job and physical safety by revealing who and where you are. For example, if you're travelling abroad and you connect to your employer's computers to check or send mail, you can inadvertently reveal your national origin and professional affiliation to anyone observing the network, even if the connection is encrypted.

How does traffic analysis work? Internet data packets have two parts: a data payload and a header used for routing. The data payload is whatever is being sent, whether that's an email message, a web page, or an audio file. Even if you encrypt the data payload of your communications, traffic analysis still reveals a great deal about what you're doing and, possibly, what you're saying. That's because it focuses on the header, which discloses source, destination, size, timing, and so on.

A basic problem for the privacy minded is that the recipient of your communications can see that you sent it by looking at headers. So can authorized intermediaries like Internet service providers, and sometimes unauthorized intermediaries as well. A very simple form of traffic analysis might involve sitting somewhere between sender and recipient on the network, looking at headers.

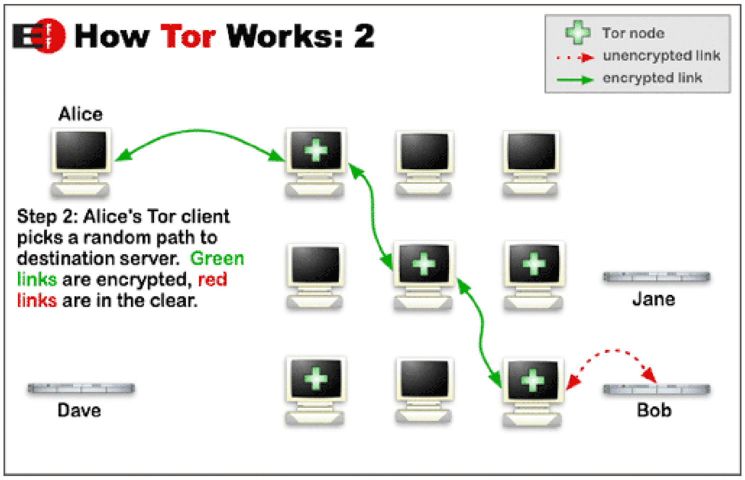
But there are also more powerful kinds of traffic analysis. Some attackers spy on multiple parts of the Internet and use sophisticated statistical techniques to track the communications patterns of many different organizations and individuals. Encryption does not help against these attackers, since it only hides the content of Internet traffic, not the headers.

The solution: a distributed, anonymous network


How Tor Works: 1
Tor node
unencrypted link
encrypted link
Alice
Step 1: Alice's Tor client obtains a list of Tor nodes from a directory server.
Jane
Dave
Bob

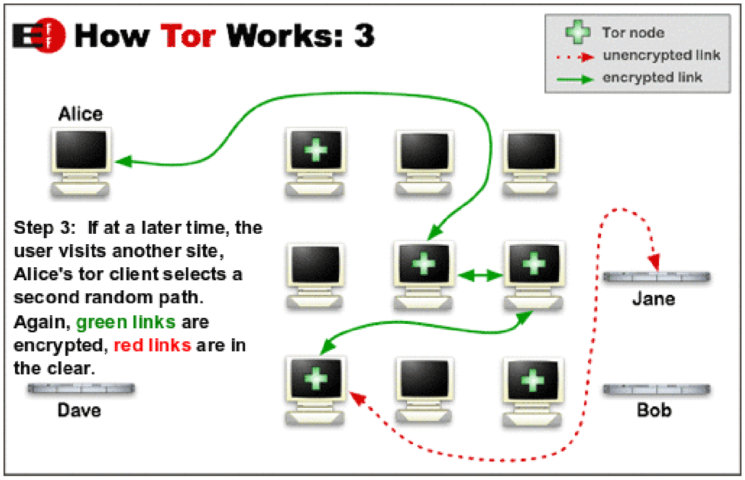
Tor helps to reduce the risks of both simple and sophisticated traffic analysis by distributing your transactions over several places on the Internet, so no single point can link you to your destination. The idea is similar to using a twisty, hard-to-follow route in order to throw off somebody who is tailing you — and then periodically erasing your footprints. Instead of taking a direct route from source to destination, data packets on the Tor network take a random pathway through several relays that cover your tracks so no observer at any single point can tell where the data came from or where it's going.

To create a private network pathway with Tor, the user's software or client incrementally builds a circuit of encrypted connections through relays on the network. The circuit is extended one hop at a time, and each relay along the way knows only which relay gave it data and which relay it is giving data to. No individual relay ever knows the complete path that a data packet has taken. The client negotiates a separate set of encryption keys for each hop along the circuit to ensure that each hop can't trace these connections as they pass through.


How Tor Works: 2
Tor node
unencrypted link
encrypted link
Alice
Step 2: Alice's Tor client picks a random path to destination server. Green links are encrypted, red links are in the clear.
Jane
Dave
Bob

Once a circuit has been established, many kinds of data can be exchanged and several different sorts of software applications can be deployed over the Tor network. Because each relay sees no more than one hop in the circuit, neither an eavesdropper nor a compromised relay can use traffic analysis to link the connection's source and destination. Tor only works for TCP streams and can be used by any application with SOCKS support.

For efficiency, the Tor software uses the same circuit for connections that happen within the same ten minutes or so. Later requests are given a new circuit, to keep people from linking your earlier actions to the new ones.


How Tor Works: 3
Tor node
unencrypted link
encrypted link
Alice
Jane
Dave
Bob
Step 3: If at a later time, the user visits another site, Alice's tor client selects a second random path. Again, green links are encrypted, red links are in the clear.

Staying anonymous

Tor can't solve all anonymity problems. It focuses only on protecting the transport of data. You need to use protocol-specific support software if you don't want the sites you visit to see your identifying information. For example, you can use Tor Browser while browsing the web to withhold some information about your computer's configuration.

Also, to protect your anonymity, be smart. Don't provide your name or other revealing information in web forms. Be aware that, like all anonymizing networks that are fast enough for web browsing, Tor does not provide protection against end-to-end timing attacks: If your attacker can watch the traffic coming out of your computer, and also the traffic arriving at your chosen destination, he can use statistical analysis to discover that they are part of the same circuit.

The future of Tor

Providing a usable anonymizing network on the Internet today is an ongoing challenge. We want software that meets users' needs. We also want to keep the network up and running in a way that handles as many users as possible. Security and usability don't have to be at odds: As Tor's usability increases, it will attract more users, which will increase the possible sources and destinations of each communication, thus increasing security for everyone. We're making progress, but we need your help. Please consider running a relay or volunteering as a developer.

Ongoing trends in law, policy, and technology threaten anonymity as never before, undermining our ability to speak and read freely online. These trends also undermine national security and critical infrastructure by making communication among individuals, organizations, corporations, and governments more vulnerable to analysis. Each new user and relay provides additional diversity, enhancing Tor's ability to put control over your security and privacy back into your hands.

Trademark, copyright notices, and rules for use by third parties can be found in our FAQ.

About Tor
What Tor Does
Users of Tor
Core Tor People
Sponsors
Contact Us

Get Involved
Donate
Mailing Lists
Mirrors
Hidden Services
Translations

Documentation
Manuals
Installation
Guides
Tor Wiki
General Tor FAQ

Proud member of THE INTERNET DEFENSE LEAGUE EST 2012

EXHIBIT 2

Documentation Overview
Installation Guides
Manuals
Tor Wiki
General FAQ
Abuse FAQ
Trademark FAQ
Tor Legal FAQ
Tor DMCA Response

Tor Tip

Tor is written for and supported by people like you. Donate today!

Tor FAQ

General questions:

Compilation and Installation:

Tor Browser (general):

Tor Browser (3.x and later):

Advanced Tor usage:

Running a Tor relay:

Tor hidden services:

Development:

Anonymity and Security:

Alternate designs that we don't do (yet):

Abuse:

For other questions not yet on this version of the FAQ, see the wiki FAQ for now.

General:

What is Tor?

The name "Tor" can refer to several different components.

The Tor software is a program you can run on your computer that helps keep you safe on the Internet. Tor protects you by bouncing your communications around a distributed network of relays run by volunteers all around the world: it prevents somebody watching your Internet connection from learning what sites you visit, and it prevents the sites you visit from learning your physical location. This set of volunteer relays is called the Tor network. You can read more about how Tor works on the overview page.

The Tor Project is a non-profit (charity) organization that maintains and develops the Tor software.

How is Tor different from other proxies?

A typical proxy provider sets up a server somewhere on the Internet and allows you to use it to relay your traffic. This creates a simple, easy to maintain architecture. The users all enter and leave through the same server. The provider may charge for use of the proxy, or fund their costs through advertisements on the server. In the simplest configuration, you don't have to install anything. You just have to point your browser at their proxy server. Simple proxy providers are fine solutions if you do not want protections for your privacy and anonymity online and you trust the provider from doing bad things. Some simple proxy providers use SSL to secure your connection to them. This may protect you against local eavesdroppers, such as those at a cafe with free wifi Internet.

Simple proxy providers also create a single point of failure. The provider knows who you are and where you browse on the Internet. They can see your traffic as it passes through their server. In some cases, they can even see inside your encrypted traffic as they relay it to your banking site or to ecommerce stores. You have to trust the provider isn't doing any number of things, such as watching your traffic, injecting their own advertisements into your traffic stream, and recording your personal details.

Tor passes your traffic through at least 3 different servers before sending it on to the destination. Because there's a separate layer of encryption for each of the three relays, Tor does not modify, or even know, what you are sending into it. It merely relays your traffic, completely encrypted through the Tor network and has it pop out somewhere else in the world, completely intact. The Tor client is required because we assume you trust your local computer. The Tor client manages the encryption and the path chosen through the network. The relays located all over the world merely pass encrypted packets between themselves.

Doesn't the first server see who I am?

Possibly. A bad first of three servers can see encrypted Tor traffic coming from your computer. It still doesn't know who you are and what you are doing over Tor. It merely sees "This IP address is using Tor". Tor is not illegal anywhere in the world, so using Tor by itself is fine. You are still protected from this node figuring out who you are and where you are going on the Internet.

Can't the third server see my traffic?

Possibly. A bad third of three servers can see the traffic you sent into Tor. It won't know who sent this traffic. If you're using encryption, such as visiting a bank or e-commerce website, or encrypted mail connections, etc, it will only know the destination. It won't be able to see the data inside the traffic stream. You are still protected from this node figuring out who you are and if using encryption, what data you're sending to the destination.

What programs can I use with Tor?

If you want to use Tor with a web browser, we provide the Tor Browser, which includes everything you need to browse the web safely using Tor. If you want to use another web browser with Tor, see Other web browsers.

There are plenty of other programs you can use with Tor, but we haven't researched the application-level anonymity issues on all of them well enough to be able to recommend a safe configuration. Our wiki has a list of instructions for Torifying specific applications. Please add to these lists and help us keep them accurate!

Why is it called Tor?

Because Tor is the onion routing network. When we were starting the new next-generation design and implementation of onion routing in 2001-2002, we would tell people we were working on onion routing, and they would say "Neat. Which one?" Even if onion routing has become a standard household term, Tor was born out of the actual onion routing project run by the Naval Research Lab.

(It's also got a fine translation from German and Turkish.)

Note: even though it originally came from an acronym, Tor is not spelled "TOR". Only the first letter is capitalized. In fact, we can usually spot people who haven't read any of our website (and have instead learned everything they know about Tor from news articles) by the fact that they spell it wrong.

Is there a backdoor in Tor?

There is absolutely no backdoor in Tor. We know some smart lawyers who say that it's unlikely that anybody will try to make us add one in our jurisdiction (U.S.). If they do ask us, we will fight them, and (the lawyers say) probably win.

We think that putting a backdoor in Tor would be tremendously irresponsible to our users, and a bad precedent for security software in general. If we ever put a deliberate backdoor in our security software, it would ruin our professional reputations. Nobody would trust our software ever again — for excellent reason!

But that said, there are still plenty of subtle attacks people might try. Somebody might impersonate us, or break into our computers, or something like that. Tor is open source, and you should always check the source (or at least the diffs since the last release) for suspicious things. If we (or the distributors) don't give you source, that's a sure sign something funny might be going on. You should also check the PGP signatures on the releases, to make sure nobody messed with the distribution sites.

Also, there might be accidental bugs in Tor that could affect your anonymity. We periodically find and fix anonymity-related bugs, so make sure you keep your Tor versions up-to-date.

Can I distribute Tor?

Yes.

The Tor software is free software. This means we give you the rights to redistribute the Tor software, either modified or unmodified, either for a fee or gratis. You don't have to ask us for specific permission.

However, if you want to redistribute the Tor software you must follow our LICENSE. Essentially this means that you need to include our LICENSE file along with whatever part of the Tor software you're distributing.

Most people who ask us this question don't want to distribute just the Tor software, though. They want to distribute the Tor Browser. This includes Firefox Extended Support Release, and the NoScript and HTTPS-Everywhere extensions. You will need to follow the license for those programs as well. Both of those Firefox extensions are distributed under the GNU General Public License, while Firefox ESR is released under the Mozilla Public License. The simplest way to obey their licenses is to include the source code for these programs everywhere you include the bundles themselves.

Also, you should make sure not to confuse your readers about what Tor is, who makes it, and what properties it provides (and doesn't provide). See our trademark FAQ for details.

Lastly, you should realize that we release new versions of the Tor software frequently, and sometimes we make backward incompatible changes. So if you distribute a particular version of the Tor software, it may not be supported — or even work — six months later. This is a fact of life for all security software under heavy development.

How can I get support?

Your best bet is to first try the following:

a. Read through this FAQ.
b. Read through the documentation.
c. Read through the tor-talk archives and see if your question is already answered.
d. Join our irc channel and state the issue and wait for help.
e. Send an email to help@rt.torproject.org.
f. If all else fails, try contacting us directly.

If you find your answer, please stick around on the IRC channel or the mailing list to help others who were once in your position.

Is there a Tor forum?

We have a StackExchange page that is currently in public beta.

Why is Tor so slow?

There are many reasons why the Tor network is currently slow.

Before we answer, though, you should realize that Tor is never going to be blazing fast. Your traffic is bouncing through volunteers' computers in various parts of the world, and some bottlenecks and network latency will always be present. You shouldn't expect to see university-style bandwidth through Tor.

But that doesn't mean that it can't be improved. The current Tor network is quite small compared to the number of people trying to use it, and many of these users don't understand or care that Tor can't currently handle file-sharing traffic load.

For the much more in-depth answer, see Roger's blog post on the topic, which includes both a detailed PDF and a video to go with it.

What can you do to help?

- Configure your Tor to relay traffic for others. Help make the Tor network large enough that we can handle all the users who want privacy and security on the Internet.
- Help us make Tor more usable. We especially need people to help make it easier to configure your Tor as a relay. Also, we need help with clear simple documentation to walk people through setting it up.
- There are some bottlenecks in the current Tor network. Help us design experiments to track down and demonstrate where the problems are, and then we can focus better on fixing them.
- Tor needs some architectural changes too. One important change is to start providing better service to people who relay traffic. We're working on this, and we'll finish faster if we get to spend more time on it.
- Help do other things so we can do the hard stuff. Please take a moment to figure out what your skills and interests are, and then look at our volunteer page.
- Help find sponsors for Tor. Do you work at a company or government agency that uses Tor or has a use for Internet privacy, e.g. to browse the competition's websites discreetly, or to connect back to the home servers when on the road without revealing affiliations? If your organization has an interest in keeping the Tor network working, please contact them about supporting Tor. Without sponsors, Tor is going to become even slower.
- If you can't help out with any of the above, you can still help out individually by donating a bit of money to the cause. It adds up!

How can I share files anonymously through Tor?

File sharing (peer-to-peer/P2P) is widely unwanted in the Tor network, and exit nodes are configured to block file sharing traffic by default. Tor is not really designed for it, and file sharing through Tor slows down everyone's browsing. Also, Bittorrent over Tor is not anonymous!

What would The Tor Project do with more funding?

The Tor network's several thousand relays push over 7.5GB per second on average. We have millions of daily users. But the Tor network is not yet self-sustaining.

There are six main development/maintenance pushes that need attention:

- Scalability: We need to keep scaling and decentralizing the Tor architecture so it can handle thousands of relays and millions of users. The upcoming stable release is a major improvement, but there's lots more to be done next in terms of keeping Tor fast and stable.
- User support: With this many users, a lot of people are asking questions all the time, offering to help out with things, and so on. We need good clean docs, and we need to spend some effort coordinating volunteers.
- Relay support: the Tor network is run by volunteers, but they still need attention with prompt bug fixes, explanations when things go wrong, reminders to upgrade, and so on. The network itself is a commons, and somebody needs to spend some energy making sure the relay operators stay happy. We also need to work on stability on some platforms — e.g., Tor relays have problems on Win XP currently.
- Usability: Beyond documentation, we also need to work on usability of the software itself. This includes installers, clean GUIs, easy configuration to interface with other applications, and generally automating all of the difficult and confusing steps inside Tor. Usability for privacy software has never been easy.
- Incentives: We need to work on ways to encourage people to configure their Tors as relays and exit nodes rather than just clients. We need to make it easy to become a relay, and we need to give people incentives to do it.
- Research: The anonymous communications field is full of surprises and gotchas. In our copious free time, we also help run top anonymity and privacy conferences like PETS. We've identified a set of critical Tor research questions that will help us figure out how to make Tor secure against the variety of attacks out there. Of course, there are more research questions waiting behind these.

We're continuing to move forward on all of these, but at this rate the Tor network is growing faster than the developers can keep up. Now would be an excellent time to add a few more developers to the effort so we can continue to grow the network.

We are also excited about tackling related problems, such as censorship-resistance.

We are proud to have sponsorship and support from the Omidyar Network, the International Broadcasting Bureau, Bell Security Solutions, the Electronic Frontier Foundation, several government agencies and research groups, and hundreds of private contributors.

However, this support is not enough to keep Tor abreast of changes in the Internet privacy landscape. Please donate to the project, or contact our executive director for information on making grants or major donations.

Can I use Tor on my phone or mobile device?

Tor on Android devices is maintained by the Guardian Project. Currently, there is no supported way of using Tor on iOS; the Guardian Project is working to make this a reality in the future.

Which outbound ports must be open when using Tor as a client?

Tor may attempt to connect to any port that is advertised in the directory as an ORPort (for making Tor connections) or a DirPort (for fetching updates to the directory). There are a variety of these ports: many of them are running on 80, 443, 9001, and 9030, but many use other ports too.

When using Tor as a client, you could probably get away with opening only those four ports. Since Tor does all its connections in the background, it will retry ones that fail, and hopefully you'll never have to know that it failed, as long as it finds a working one often enough. However, to get the most diversity in your entry nodes — and thus the most security — as well as the most robustness in your connectivity, you'll want to let it connect to all of them. See the FAQ entry on firewalled ports if you want to explicitly tell your Tor client which ports are reachable for you.

How can I tell if Tor is working, and that my connections really are anonymized?

There are sites you can visit that will tell you if you appear to be coming through the Tor network. Try the Tor Check site and see whether it thinks you are using Tor or not.

How do I use my browser for ftp with Tor?

Use Tor Browser. If you want a separate application for an ftp client, we've heard good things about FileZilla for Windows. You can configure it to point to Tor as a "socks4a" proxy on "localhost" port "9050".

Does Tor remove personal information from the data my application sends?

No, it doesn't. You need to use a separate program that understands your application and protocol and knows how to clean or "scrub" the data it sends. The Tor Browser tries to keep application-level data, like the user-agent string, uniform for all users. The Tor Browser can't do anything about text that you type into forms, though. Be careful and be smart.

How many people use Tor? How many relays or exit nodes are there?

All this and more about measuring Tor can be found at the Tor Metrics Portal.

What are the SSL certificate fingerprints for Tor's various websites?

*.torproject.org SSL certificate from Digicert:

```
Issued Certificate
Version: 3
Serial Number: 09 48 B1 A9 3B 25 1D 0D B1 05 10 59 E2 C2 68 0A
Not Valid Before: 2013-10-22
Not Valid After: 2016-05-03
Certificate Fingerprints
SHA1: 84 24 56 56 8E D7 90 43 47 AA 89 AB 77 7D A4 94 3B A1 A7 D5
MD5: A4 16 66 80 AE B9 A4 EC AA 88 01 1B 6F B9 EB CB
```

blog.torproject.org SSL certificate from RapidSSL:

```
Issued Certificate
Version: 3
Serial Number: 05 CA 2A A9 A5 D6 ED 44 C7 2D 88 1A 18 B0 E7 DC
Not Valid Before: 2014-04-09
Not Valid After: 2017-06-14
Certificate Fingerprints
SHA1: DE 20 3D 46 FD C3 68 EB BA 40 56 39 F5 FA FD F5 4E 3A 1F 83
MD5: 8A 8A A2 5E D9 7F 84 4C 8F 00 3B 43 E0 2D E6 4D
```

Compilation And Installation:

How do I uninstall Tor?

Tor Browser does not install itself in the classic sense of applications. You just simply delete the folder or directory named "Tor Browser" and it is removed from your system.

If this is not related to Tor Browser, uninstallation depends entirely on how you installed it and which operating system you have. If you installed a package, then hopefully your package has a way to uninstall itself. The Windows packages include uninstallers.

For Mac OS X, follow the uninstall directions.

If you installed by source, I'm afraid there is no easy uninstall method. But on the bright side, by default it only installs into /usr/local/ and it should be pretty easy to notice things there.

What are these "sig" files on the download page?

These are PGP signatures, so you can verify that the file you've downloaded is exactly the one that we intended you to get.

Please read the verifying signatures page for details.

Your website is blocked in my country. How do I download Tor?

Some government or corporate firewalls censor connections to Tor's website. In those cases, you have three options. First, get it from a friend — Tor Browser fits nicely on a USB key. Second, find the google cache for the Tor mirrors page and see if any of those copies of our website work for you. Third, you can download Tor Browser via email: log in to your email account and send an email to '`gettor@torproject.org`' with one of the following words in the body of the message: `windows`, `osx` or `linux` (case insensitive). You will receive a reply with links from popular cloud services to download Tor Browser for Windows, Mac OS X or Linux, depending on the option you chose. Currently, the only cloud service supported is Dropbox. If you send a blank message or anything different from the options mentioned, you will receive a help message with detailed instructions to ask for Tor Browser via email. Please note that you can use this service from any email address: gmail, yahoo, hotmail, riseup, etc. The only restriction is that you can do a maximum of three requests in a row, after that you'll have to wait 20 minutes to use it again. See the GetTor section for more information.

Be sure to verify the signature of any package you download, especially when you get it from somewhere other than our official HTTPS website.

Why does my Tor executable appear to have a virus or spyware?

Sometimes, overzealous Windows virus and spyware detectors trigger on some parts of the Tor Windows binary. Our best guess is that these are false positives — after all, the anti-virus and anti-spyware business is just a guessing game anyway. You should contact your vendor and explain that you have a program that seems to be triggering false positives. Or pick a better vendor.

In the meantime, we encourage you to not just take our word for it. Our job is to provide the source; if you're concerned, please do recompile it yourself.

How do I open a .tar.gz or .tar.xz file?

Tar is a common archive utility for Unix and Linux systems. If your system has a mouse, you can usually open them by double clicking. Otherwise open a command prompt and execute

```
tar xzf <FILENAME>.tar.gz
```

or

```
tar xJf <FILENAME>.tar.xz
```

as documented on tar's man page.

Is there a LiveCD or other bundle that includes Tor?

Yes. Use The Amnesic Incognito Live System or Tor Browser.

Tor Browser (general):

Why can't I view videos on some Flash-based sites?

Some sites require third party browser plugins such as Flash. Plugins operate independently from Firefox and can perform activity on your computer that ruins your anonymity. This includes but is not limited to: completely disregarding proxy settings, querying your local IP address, and storing their own cookies. It is possible to use a LiveCD solution such as or The Amnesic Incognito Live System that creates a secure, transparent proxy to protect you from proxy bypass, however issues with local IP address discovery and Flash cookies still remain.

I'm using Ubuntu and I can't start Tor Browser.

You'll need to tell Ubuntu that you want the ability to execute shell scripts from the graphical interface. Open "Files" (Unity's explorer), open Preferences-> Behavior Tab -> Set "Run executable text files when they are opened" to "Ask every time", then OK.

You can also start the Tor Browser from the command line by running

```
./start-tor-browser
```

from inside the Tor Browser directory.

I'm using the Sophos anti-virus software on my Mac, and Tor starts but I can't browse anywhere.

You'll need to modify Sophos anti-virus so that Tor can connect to the internet. Go to Preferences -> Web Protection -> General, and turn off the protections for "Malicious websites" and "Malicious downloads".

We encourage affected Sophos users to contact Sophos support about this issue.

When I start Tor Browser I get an error message: "Cannot load XPCOM".

This problem is specifically caused by the Webroot SecureAnywhere Antivirus software. From the Webroot control panel, go to Identity Protection → Application Protection, and set all the files in your Tor Browser folder to 'Allow'. We encourage affected Webroot users to contact Webroot support about this issue.

Can I install other Firefox extensions?

The Tor Browser is free software, so there is nothing preventing you from modifying it any way you like. However, we do not recommend installing any additional Firefox add-ons with Tor Browser. Add-ons can break your anonymity in a number of ways, including browser fingerprinting and bypassing proxy settings.

Some people have suggested we include ad-blocking software or anti-tracking software with the Tor Browser. Right now, we do not think that's such a good idea. Tor Browser aims to provide sufficient privacy that additional add-ons to stop ads and trackers are not necessary. Using add-ons like these may cause some sites to break, which we don't want to do. Additionally, maintaining a list of "bad" sites that should be black-listed provides another opportunity to uniquely fingerprint users.

Why is NoScript configured to allow JavaScript by default in Tor Browser? Isn't that unsafe?

We configure NoScript to allow JavaScript by default in Tor Browser because many websites will not work with JavaScript disabled. Most users would give up on Tor entirely if a website they want to use requires JavaScript, because they would not know how to allow a website to use JavaScript (or that enabling JavaScript might make a website work).

There's a tradeoff here. On the one hand, we should leave JavaScript enabled by default so websites work the way users expect. On the other hand, we should disable JavaScript by default to better protect against browser vulnerabilities (not just a theoretical concern!). But there's a third issue: websites can easily determine whether you have allowed JavaScript for them, and if you disable JavaScript by default but then allow a few websites to run scripts (the way most people use NoScript), then your choice of whitelisted websites acts as a sort of cookie that makes you recognizable (and distinguishable), thus harming your anonymity.

Ultimately, we want the default Tor bundles to use a combination of firewalls (like the iptables rules in Tails) and sandboxes to make JavaScript not so scary. In the shorter term, TBB 3.0 will hopefully allow users to choose their JavaScript settings more easily — but the partitioning concern will remain.

Until we get there, feel free to leave JavaScript on or off depending on your security, anonymity, and usability priorities.

I want to use Chrome/IE/Opera/etc with Tor.

In short, using any browser besides Tor Browser with Tor is a really bad idea.

Our efforts to work with the Chrome team to add missing APIs were unsuccessful, unfortunately. Currently, it is impossible to use other browsers and get the same level of protections as when using the Tor Browser.

Google makes me solve a CAPTCHA or tells me I have spyware installed.

This is a known and intermittent problem; it does not mean that Google considers Tor to be spyware.

When you use Tor, you are sending queries through exit relays that are also shared by thousands of other users. Tor users typically see this message when many Tor users are querying Google in a short period of time. Google interprets the high volume of traffic from a single IP address (the exit relay you happened to pick) as somebody trying to "crawl" their website, so it slows down traffic from that IP address for a short time.

An alternate explanation is that Google tries to detect certain kinds of spyware or viruses that send distinctive queries to Google Search. It notes the IP addresses from which those queries are received (not realizing that they are Tor exit relays), and tries to warn any connections coming from those IP addresses that recent queries indicate an infection.

To our knowledge, Google is not doing anything intentionally specifically to deter or block Tor use. The error message about an infected machine should clear up again after a short time.

Why does Google show up in foreign languages?

Google uses "geolocation" to determine where in the world you are, so it can give you a personalized experience. This includes using the language it thinks you prefer, and it also includes giving you different results on your queries.

If you really want to see Google in English you can click the link that provides that. But we consider this a feature with Tor, not a bug --- the Internet is not flat, and it in fact does look different depending on where you are. This feature reminds people of this fact.

Note that Google search URLs take name/value pairs as arguments and one of those names is "hl". If you set "hl" to "en" then Google will return search results in English regardless of what Google server you have been sent to. On a query this looks like:

```
https://encrypted.google.com/search?q=online%20anonymity&hl=en
```

Another method is to simply use your country code for accessing Google. This can be google.be, google.de, google.us and so on.

Gmail warns me that my account may have been compromised.

Sometimes, after you've used Gmail over Tor, Google presents a pop-up notification that your account may have been compromised. The notification window lists a series of IP addresses and locations throughout the world recently used to access your account.

In general this is a false alarm: Google saw a bunch of logins from different places, as a result of running the service via Tor, and decided it was a good idea to confirm the account was being accessed by it's rightful owner.

Even though this may be a biproduct of using the service via tor, that doesn't mean you can entirely ignore the warning. It is *probably* a false positive, but it might not be since it is possible for someone to hijack your Google cookie.

Cookie hijacking is possible by either physical access to your computer or by watching your network traffic. In theory only physical access should compromise your system because Gmail and similar services should only send the cookie over an SSL link. In practice, alas, it's way more complex than that.

And if somebody *did* steal your google cookie, they might end up logging in from unusual places (though of course they also might not). So the summary is that since you're using Tor, this security measure that Google uses isn't so useful for you, because it's full of false positives. You'll have to use other approaches, like seeing if anything looks weird on the account, or looking at the timestamps for recent logins and wondering if you actually logged in at those times.

My internet connection requires an HTTP or SOCKS Proxy

You can set Proxy IP address, port, and authentication information in Tor Browser's Network Settings. If you're using Tor another way, check out the HTTPProxy and HTTPSProxy config options in the man page, and modify your torrc file accordingly. You will need an HTTP proxy for doing GET requests to fetch the Tor directory, and you will need an HTTPS proxy for doing CONNECT requests to get to Tor relays. (It's fine if they're the same proxy.) Tor also recognizes the torrc options Socks4Proxy and Socks5Proxy.

Also read up on the HTTPProxyAuthenticator and HTTPSProxyAuthenticator options if your proxy requires auth. We only support basic auth currently, but if you need NTLM authentication, you may find this post in the archives useful.

If your proxies only allow you to connect to certain ports, look at the entry on Firewalled clients for how to restrict what ports your Tor will try to access.

I want to run another application through Tor.

If you are trying to use some external application with Tor, step zero should be to reread the set of warnings for ways you can screw up. Step one should be to try to use a SOCKS proxy rather than an HTTP proxy. Typically Tor listens for SOCKS connections on port 9050. Tor Browser listens on port 9150.

If your application doesn't support SOCKS proxies, feel free to install privoxy. However, please realize that this approach is not recommended for novice users. Privoxy has an example configuration of Tor and Privoxy.

If you're unable to use the application's native proxy settings, all hope is not lost. See below.

What should I do if I can't set a proxy with my application?

On Unix, we recommend you give torsocks a try. Alternative proxifying tools like socat and proxychains are also available.

The Windows way to force applications through Tor is less clear. Some tools have been proposed , but we'd also like to see further testing done here.

Tor Browser (3.x and later):

Where did the world map (Vidalia) go?

Vidalia has been replaced with Tor Launcher, which is a Firefox extension that provides similar functionality. Unfortunately, circuit status reporting is still missing, but we are working on providing it.

How do I disable JavaScript?

Alas, Mozilla decided to get rid of the config checkbox for JavaScript from earlier Firefox versions. And since TBB 3.5 is based on Firefox 24 (FF17 is unmaintained), that means TBB 3.5 doesn't have the config checkbox anymore either, which is unfortunate.

The simplest way to disable JavaScript in TBB 3.5 is to click on the Noscript "S" (between the green onion and the address bar), and select "Forbid scripts globally". Note that vanilla NoScript actually whitelists several domains even when you try to disable scripts globally, whereas Tor Browser's NoScript configuration disables all of them.

The more klunky way to disable JavaScript is to go to about:config, find javascript.enabled, and set it to false.

There is also a very simple addon available at addons.mozilla.org called QuickJS, which provides a toolbar toggle for the javascript.enabled about:config control. There are no configuration options for the addon, it just switches the javascript.enabled entry between true and false and provides a button for it.

If you want to be extra safe, use both the about:config setting and NoScript.

As for whether you should disable it or leave it enabled, that's a tradeoff we leave to you.

How do I verify the download (sha256sums.txt)?

Instructions are on the verifying signatures page.

Why does "New Identity" close all my open tabs?

That's actually a feature, since it's discarding your application-level browser data too.

We're working on ways to make the behavior less surprising, e.g. a popup warning or auto restoring tabs. See ticket #9906 and ticket #10400 to follow progress there.

How do I configure Tor as a relay or bridge?

You've got three options.

First (best option), if you're on Linux, you can install the system Tor package (e.g. apt-get install tor) and then set it up to be a relay (instructions). You can then use TBB independent of that.

Second (complex option), you can edit your torrc file (in Data/Tor/torrc) directly to add the following lines:

```
ORPort 443
Exitpolicy reject *:*
BridgeRelay 1  # only add this line if you want to be a bridge
```

If you've installed Obfsproxy, you'll need to add one more line:

```
ServerTransportPlugin obfs3 exec /usr/bin/obfsproxy managed
```

Why are the file timestamps from 2000?

One of the huge new features in TBB 3.x is the "deterministic build" process, which allows many people to build Tor Browser and verify that they all make exactly the same package. See Mike's first blog post for the motivation, and his second blog post for the technical details of how we do it.

Part of creating identical builds is having everybody use the same timestamp. Mike picked the beginning of 2000 for that time. The reason you might see 7pm in 1999 is because of time zones.

Where is the source code for Tor Browser? How do I verify a build?

Start with https://gitweb.torproject.org/builders/tor-browser-bundle.git and https://gitweb.torproject.org/builders/tor-browser-bundle.git/tree/gitian/README.build.

Advanced Tor usage:

I'm supposed to "edit my torrc". What does that mean?

Tor installs a text file called torrc that contains configuration instructions for how your Tor program should behave. The default configuration should work fine for most Tor users.

If you installed Tor Browser, look for `Browser/TorBrowser/Data/Tor/torrc` inside your Tor Browser directory. On OS X, you must right-click or command-click on the Tor Browser icon, and select "Show Package Contents" before the Tor Browser directories become visible.

Tor puts the torrc file in `/usr/local/etc/tor/torrc` if you compiled tor from source, and `/etc/tor/torrc` or `/etc/torrc` if you installed a pre-built package.

Once you've changed your torrc, you will need to restart tor for the changes to take effect. (For advanced users, note that you actually only need to send Tor a HUP signal, not actually restart it.)

For other configuration options you can use, see the Tor manual page. Have a look at the sample torrc file for hints on common configurations. Remember, all lines beginning with # in torrc are treated as comments and have no effect on Tor's configuration.

How do I set up logging, or see Tor's logs?

You'll have to go find the log files by hand. Here are some likely places for your logs to be:

- On OS X, Debian, Red Hat, etc, the logs are in /var/log/tor/
- On Windows, there are no default log files currently. If you enable logs in your torrc file, they default to `\username\Application Data\tor\log\` or `\Application Data\tor\log\`
- If you compiled Tor from source, by default your Tor logs to "stdout" at log-level notice. If you enable logs in your torrc file, they default to `/usr/local/var/log/tor/`.

To change your logging setup by hand, edit your torrc and find the section (near the top of the file) which contains the following line:

```
## Logs go to stdout at level "notice" unless redirected by something
## else, like one of the below lines.
```

For example, if you want Tor to send complete debug, info, notice, warn, and err level messages to a file, append the following line to the end of the section:

```
Log debug file c:/program files/tor/debug.log
```

Replace `c:/program files/tor/debug.log` with a directory and filename for your Tor log.

What log level should I use?

There are five log levels (also called "log severities") you might see in Tor's logs:

- "err": something bad just happened, and we can't recover. Tor will exit.

- "warn": something bad happened, but we're still running. The bad thing might be a bug in the code, some other Tor process doing something unexpected, etc. The operator should examine the message and try to correct the problem.
- "notice": something the operator will want to know about.
- "info": something happened (maybe bad, maybe ok), but there's nothing you need to (or can) do about it.
- "debug": for everything louder than info. It is quite loud indeed.

Alas, some of the warn messages are hard for ordinary users to correct -- the developers are slowly making progress at making Tor automatically react correctly for each situation.

We recommend running at the default, which is "notice". You will hear about important things, and you won't hear about unimportant things.

Tor relays in particular should avoid logging at info or debug in normal operation, since they might end up recording sensitive information in their logs.

I installed Tor but it's not working.

Once you've got Tor Browser up and running, the first question to ask is whether your Tor client is able to establish a circuit.

If Tor can establish a circuit, Tor Browser will automatically launch the browser for you. You can also check in the Tor logs for a line saying that Tor "has successfully opened a circuit. Looks like client functionality is working."

If Tor can't establish a circuit, here are some hints:

a. Check your system clock. If it's more than a few hours off, Tor will refuse to build circuits. For Microsoft Windows users, synchronize your clock under the clock -> Internet time tab. In addition, correct the day and date under the 'Date & Time' Tab. Also make sure your time zone is correct.
b. Is your Internet connection firewalled by port, or do you normally need to use a proxy?
c. Are you running programs like Norton Internet Security or SELinux that block certain connections, even though you don't realize they do? They could be preventing Tor from making network connections.
d. Are you in China, or behind a restrictive corporate network firewall that blocks the public Tor relays? If so, you should learn about Tor bridges.
e. Check your Tor logs. Do they give you any hints about what's going wrong?

My Tor keeps crashing.

We want to hear from you! There are supposed to be zero crash bugs in Tor. This FAQ entry describes the best way for you to be helpful to us. But even if you can't work out all the details, we still want to hear about it, so we can help you track it down.

First, make sure you're using the latest version of Tor (either the latest stable or the latest development version).

Second, make sure your version of libevent is new enough. We recommend at least libevent 1.3a.

Third, see if there's already an entry for your bug in the Tor bugtracker. If so, check if there are any new details that you can add.

Fourth, is the crash repeatable? Can you cause the crash? Can you isolate some of the circumstances or config options that make it happen? How quickly or often does the bug show up? Can you check if it happens with other versions of Tor, for example the latest stable release?

Fifth, what sort of crash do you get?

- Does your Tor log include an "assert failure"? If so, please tell us that line, since it helps us figure out what's going on. Tell us the previous couple of log messages as well, especially if they seem important.
- If it says "Segmentation fault - core dumped" then you need to do a bit more to track it down. Look for a file like "core" or "tor.core" or "core.12345" in your current directory, or in your Data Directory. If it's there, run "gdb tor core" and then "bt", and include the output. If you can't find a core, run "ulimit -c unlimited", restart Tor, and try to make it crash again. (This core thing will only work on Unix -- alas, tracking down bugs on Windows is harder. If you're on Windows, can you get somebody to duplicate your bug on Unix?)
- If Tor simply vanishes mysteriously, it probably is a segmentation fault but you're running Tor in the background (as a daemon) so you won't notice. Go look at the end of your log file, and look for a core file as above. If you don't find any good hints, you should consider running Tor in the foreground (from a shell) so you can see how it dies. Warning: if you switch to running Tor in the foreground, you might start using a different torrc file, with a different default Data Directory; see the relay-upgrade FAQ entry for details.
- If it's still vanishing mysteriously, perhaps something else is killing it? Do you have resource limits (ulimits) configured that kill off processes sometimes? (This is especially common on OpenBSD.) On Linux, try running "dmesg" to see if the out-of-memory killer removed your process. (Tor will exit cleanly if it notices that it's run out

of memory, but in some cases it might not have time to notice.) In very rare circumstances, hardware problems could also be the culprit.

Sixth, if the above ideas don't point out the bug, consider increasing your log level to "loglevel debug". You can look at the log-configuration FAQ entry for instructions on what to put in your torrc file. If it usually takes a long time for the crash to show up, you will want to reserve a whole lot of disk space for the debug log. Alternatively, you could just send debug-level logs to the screen (it's called "stdout" in the torrc), and then when it crashes you'll see the last couple of log lines it had printed. (Note that running with verbose logging like this will slow Tor down considerably, and note also that it's generally not a good idea security-wise to keep logs like this sitting around.)

Can I control which nodes (or country) are used for entry/exit?

Yes. You can set preferred entry and exit nodes as well as inform Tor which nodes you do not want to use. The following options can be added to your config file "torrc" or specified on the command line:

`EntryNodes $fingerprint,$fingerprint,...`

A list of preferred nodes to use for the first hop in the circuit, if possible.

`ExitNodes $fingerprint,$fingerprint,...`

A list of preferred nodes to use for the last hop in the circuit, if possible.

`ExcludeNodes $fingerprint,$fingerprint,...`

A list of nodes to never use when building a circuit.

`ExcludeExitNodes $fingerprint,$fingerprint,...`

A list of nodes to never use when picking an exit. Nodes listed in `ExcludeNodes` are automatically in this list.

We recommend you do not use these — they are intended for testing and may disappear in future versions. You get the best security that Tor can provide when you leave the route selection to Tor; overriding the entry / exit nodes can mess up your anonymity in ways we don't understand.

Note also that not every circuit is used to deliver traffic outside of the Tor network. It is normal to see non-exit circuits (such as those used to connect to hidden services, those that do directory fetches, those used for relay reachability self-tests, and so on) that end at a non-exit node. To keep a node from being used entirely, see `ExcludeNodes` and `StrictNodes` in the manual.

Instead of `$fingerprint` you can also specify a 2 letter ISO3166 country code in curly braces (for example `{de}`), or an ip address pattern (for example 255.254.0.0/8). Make sure there are no spaces between the commas and the list items.

If you want to access a service directly through Tor's Socks interface (eg. using ssh via connect.c), another option is to set up an internal mapping in your configuration file using `MapAddress`. See the manual page for details.

My firewall only allows a few outgoing ports.

If your firewall works by blocking ports, then you can tell Tor to only use the ports when you start your Tor Browser. Or you can add the ports that your firewall permits by adding "FascistFirewall 1" to your torrc configuration file. By default, when you set this Tor assumes that your firewall allows only port 80 and port 443 (HTTP and HTTPS respectively). You can select a different set of ports with the FirewallPorts torrc option.

If you want to be more fine-grained with your controls, you can also use the ReachableAddresses config options, e.g.:

```
ReachableDirAddresses *:80
ReachableORAddresses *:443
```

Is there a list of default exit ports?

The default open ports are listed below but keep in mind that, any port or ports can be opened by the relay operator by configuring it in torrc or modifying the source code. But the default according to src/or/policies.c from the source code release tor-0.2.4.16-rc is:

```
reject 0.0.0.0/8
reject 169.254.0.0/16
reject 127.0.0.0/8
reject 192.168.0.0/16
reject 10.0.0.0/8
reject 172.16.0.0/12
reject *:25
reject *:119
reject *:135-139
reject *:445
reject *:563
reject *:1214
reject *:4661-4666
reject *:6346-6429
reject *:6699
reject *:6881-6999
accept *:*
```

A relay will block access to its own IP address, as well local network IP addresses. A relay always blocks itself by default. This prevents Tor users from accidentally accessing any of the exit operator's local services.

I keep seeing these warnings about SOCKS and DNS information leaks. Should I worry?

The warning is:

Your application (using socks5 on port %d) is giving Tor only an IP address. Applications that do DNS resolves themselves may leak information. Consider using Socks4A (e.g. via Polipo or socat) instead.

If you are running Tor to get anonymity, and you are worried about an attacker who is even slightly clever, then yes, you should worry. Here's why.

The Problem. When your applications connect to servers on the Internet, they need to resolve hostnames that you can read (like www.torproject.org) into IP addresses that the Internet can use (like 209.237.230.66). To do this, your application sends a request to a DNS server, telling it the hostname it wants to resolve. The DNS server replies by telling your application the IP address.

Clearly, this is a bad idea if you plan to connect to the remote host anonymously: when your application sends the request to the DNS server, the DNS server (and anybody else who might be watching) can see what hostname you are asking for. Even if your application then uses Tor to connect to the IP anonymously, it will be pretty obvious that the user making the anonymous connection is probably the same person who made the DNS request.

Where SOCKS comes in. Your application uses the SOCKS protocol to connect to your local Tor client. There are 3 versions of SOCKS you are likely to run into: SOCKS 4 (which only uses IP addresses), SOCKS 5 (which usually uses IP addresses in practice), and SOCKS 4a (which uses hostnames).

When your application uses SOCKS 4 or SOCKS 5 to give Tor an IP address, Tor guesses that it 'probably' got the IP address non-anonymously from a DNS server. That's why it gives you a warning message: you probably aren't as anonymous as you think.

So what can I do? We describe a few solutions below.

- If your application speaks SOCKS 4a, use it.
- If you only need one or two hosts, or you are good at programming, you may be able to get a socks-based port-forwarder like socat to work for you; see the Torify HOWTO for examples.
- Tor ships with a program called tor-resolve that can use the Tor network to look up hostnames remotely; if you resolve hostnames to IPs with tor-resolve, then pass the IPs to your applications, you'll be fine. (Tor will still give the warning, but now you know what it means.)

If you think that you applied one of the solutions properly but still experience DNS leaks please verify there is no third-party application using DNS independently of Tor. Please see the FAQ entry on whether you're really absolutely anonymous using Tor for some examples.

How do I check if my application that uses SOCKS is leaking DNS requests?

These are two steps you need to take here. The first is to make sure that it's using the correct variant of the SOCKS protocol, and the second is to make sure that there aren't other leaks.

Step one: add "TestSocks 1" to your torrc file, and then watch your logs as you use your application. Tor will then log, for each SOCKS connection, whether it was using a 'good' variant or a 'bad' one. (If you want to automatically disable all 'bad' variants, set "SafeSocks 1" in your torrc file.)

Step two: even if your application is using the correct variant of the SOCKS protocol, there is still a risk that it could be leaking DNS queries. This problem happens in Firefox extensions that resolve the destination hostname themselves, for example to show you its IP address, what country it's in, etc. These applications may use a safe SOCKS variant when actually making connections, but they still do DNS resolves locally. If you suspect your application might behave like this, you should use a network sniffer like Wireshark and look for suspicious outbound DNS requests. I'm afraid the details of how to look for these problems are beyond the scope of a FAQ entry though -- find a friend to help if you have problems.

Running a Tor relay:

How do I decide if I should run a relay?

We're looking for people with reasonably reliable Internet connections, that have at least 250 kilobytes/second each way. If that's you, please consider helping out.

Why isn't my relay being used more?

If your relay is relatively new then give it time. Tor decides which relays it uses heuristically based on reports from Bandwidth Authorities. These authorities take measurements of your relay's capacity and, over time, directs more traffic there until it reaches an optimal load. The lifecycle of a new relay is explained in more depth in this blog post.

If you've been running a relay for a while and still having issues then try asking on the tor-relays list.

I don't have a static IP.

Tor can handle relays with dynamic IP addresses just fine. Just leave the "Address" line in your torrc blank, and Tor will guess.

Why do I get portscanned more often when I run a Tor relay?

If you allow exit connections, some services that people connect to from your relay will connect back to collect more information about you. For example, some IRC servers connect back to your identd port to record which user made the connection. (This doesn't really work for them, because Tor doesn't know this information, but they try anyway.) Also, users exiting from you might attract the attention of other users on the IRC server, website, etc. who want to know more about the host they're relaying through.

Another reason is that groups who scan for open proxies on the Internet have learned that sometimes Tor relays expose their socks port to the world. We recommend that you bind your socksport to local networks only.

In any case, you need to keep up to date with your security. See this article on operational security for Tor relays for more suggestions.

How can I get Tor to fully make use of my high capacity connection?

See this tor-relays thread.

How stable does my relay need to be?

We aim to make setting up a Tor relay easy and convenient:

- Tor has built-in support for rate limiting. Further, if you have a fast link but want to limit the number of bytes per day (or week or month) that you donate, check out the hibernation feature.
- Each Tor relay has an exit policy that specifies what sort of outbound connections are allowed or refused from that relay. If you are uncomfortable allowing people to exit from your relay, you can set it up to only allow connections to other Tor relays.
- It's fine if the relay goes offline sometimes. The directories notice this quickly and stop advertising the relay. Just try to make sure it's not too often, since connections using the relay when it disconnects will break.
- We can handle relays with dynamic IPs just fine — simply leave the Address config option blank, and Tor will try to guess.
- If your relay is behind a NAT and it doesn't know its public IP (e.g. it has an IP of 192.168.x.y), you'll need to set up port forwarding. Forwarding TCP connections is system dependent but this FAQ entry offers some examples on how to do this.
- Your relay will passively estimate and advertise its recent bandwidth capacity, so high-bandwidth relays will attract more users than low-bandwidth ones. Therefore having low-bandwidth relays is useful too.

How should I configure the outgoing filters on my relay?

All *outgoing* connections must be allowed, so that each relay can communicate with every other relay.

In many jurisdictions, Tor relay operators are legally protected by the same *common carrier* regulations that prevent internet service providers from being held liable for third-party content that passes through their network. Exit relays that filter some traffic would likely forfeit those protections.

Tor promotes free network access without interference. Exit relays must not filter the traffic that passes through them to the internet. Exit relays found to be filtering traffic will get the BadExit flag once detected.

What bandwidth shaping options are available to Tor relays?

There are two options you can add to your torrc file:

- BandwidthRate is the maximum long-term bandwidth allowed (bytes per second). For example, you might want to choose "BandwidthRate 10 MBytes" for 10 megabytes per second (a fast connection), or "BandwidthRate 500 KBytes" for 500 kilobytes per second (a decent cable connection). The minimum BandwidthRate setting is 20 kilobytes per second.
- BandwidthBurst is a pool of bytes used to fulfill requests during short periods of traffic above BandwidthRate but still keeps the average over a long period to BandwidthRate. A low Rate but a high Burst enforces a long-term average while still allowing more traffic during peak times if the average hasn't been reached lately. For example, if you choose "BandwidthBurst 500 KBytes" and also use that for your BandwidthRate, then you will never use more

than 500 kilobytes per second; but if you choose a higher BandwidthBurst (like 5 MBytes), it will allow more bytes through until the pool is empty.

If you have an asymmetric connection (upload less than download) such as a cable modem, you should set BandwidthRate to less than your smaller bandwidth (Usually that's the upload bandwidth). (Otherwise, you could drop many packets during periods of maximum bandwidth usage -- you may need to experiment with which values make your connection comfortable.) Then set BandwidthBurst to the same as BandwidthRate.

Linux-based Tor nodes have another option at their disposal: they can prioritize Tor traffic below other traffic on their machine, so that their own personal traffic is not impacted by Tor load. A script to do this can be found in the Tor source distribution's contrib directory.

Additionally, there are hibernation options where you can tell Tor to only serve a certain amount of bandwidth per time period (such as 100 GB per month). These are covered in the hibernation entry below.

Note that BandwidthRate and BandwidthBurst are in **Bytes**, not Bits.

How can I limit the total amount of bandwidth used by my Tor relay?

The accounting options in the torrc file allow you to specify the maximum amount of bytes your relay uses for a time period.

```
    AccountingStart day week month [day] HH:MM
```

This specifies when the accounting should reset. For instance, to setup a total amount of bytes served for a week (that resets every Wednesday at 10:00am), you would use:

```
    AccountingStart week 3 10:00
    AccountingMax 500 GBytes
```

This specifies the maximum amount of data your relay will send during an accounting period, and the maximum amount of data your relay will receive during an account period. When the accounting period resets (from AccountingStart), then the counters for AccountingMax are reset to 0.

Example: Let's say you want to allow 50 GB of traffic every day in each direction and the accounting should reset at noon each day:

```
    AccountingStart day 12:00
    AccountingMax 50 GBytes
```

Note that your relay won't wake up exactly at the beginning of each accounting period. It will keep track of how quickly it used its quota in the last period, and choose a random point in the new interval to wake up. This way we avoid having hundreds of relays working at the beginning of each month but none still up by the end.

If you have only a small amount of bandwidth to donate compared to your connection speed, we recommend you use daily accounting, so you don't end up using your entire monthly quota in the first day. Just divide your monthly amount by 30. You might also consider rate limiting to spread your usefulness over more of the day: if you want to offer X GB in each direction, you could set your RelayBandwidthRate to 20*X KBytes. For example, if you have 50 GB to offer each way, you might set your RelayBandwidthRate to 1000 KBytes: this way your relay will always be useful for at least half of each day.

```
    AccountingStart day 0:00
    AccountingMax 50 GBytes
    RelayBandwidthRate 1000 KBytes
    RelayBandwidthBurst 5000 KBytes # allow higher bursts but maintain average
```

Why does my relay write more bytes onto the network than it reads?

You're right, for the most part a byte into your Tor relay means a byte out, and vice versa. But there are a few exceptions:

If you open your DirPort, then Tor clients will ask you for a copy of the directory. The request they make (an HTTP GET) is quite small, and the response is sometimes quite large. This probably accounts for most of the difference between your "write" byte count and your "read" byte count.

Another minor exception shows up when you operate as an exit node, and you read a few bytes from an exit connection (for example, an instant messaging or ssh connection) and wrap it up into an entire 512 byte cell for transport through the Tor network.

Why can I not browse anymore after limiting bandwidth on my Tor relay?

The parameters assigned in the AccountingMax and BandwidthRate apply to both client and relay functions of the Tor process. Thus you may find that you are unable to browse as soon as your Tor goes into hibernation, signaled by this entry in the log:

```
Bandwidth soft limit reached; commencing hibernation. No new
    connections will be accepted
```

The solution is to run two Tor processes - one relay and one client, each with its own config. One way to do this (if you are starting from a working relay setup) is as follows:

- In the relay Tor torrc file, simply set the SocksPort to 0.
- Create a new client torrc file from the torrc.sample and ensure it uses a different log file from the relay. One naming convention may be torrc.client and torrc.relay.
- Modify the Tor client and relay startup scripts to include '-f /path/to/correct/torrc'.
- In Linux/BSD/Mac OS X, changing the startup scripts to Tor.client and Tor.relay may make separation of configs easier.

I'd run a relay, but I don't want to deal with abuse issues.

Great. That's exactly why we implemented exit policies.

Each Tor relay has an exit policy that specifies what sort of outbound connections are allowed or refused from that relay. The exit policies are propagated to Tor clients via the directory, so clients will automatically avoid picking exit relays that would refuse to exit to their intended destination. This way each relay can decide the services, hosts, and networks he wants to allow connections to, based on abuse potential and his own situation. Read the FAQ entry on issues you might encounter if you use the default exit policy, and then read Mike Perry's tips for running an exit node with minimal harassment.

The default exit policy allows access to many popular services (e.g. web browsing), but restricts some due to abuse potential (e.g. mail) and some since the Tor network can't handle the load (e.g. default file-sharing ports). You can change your exit policy by editing your torrc file. If you want to avoid most if not all abuse potential, set it to "reject *:*". This setting means that your relay will be used for relaying traffic inside the Tor network, but not for connections to external websites or other services.

If you do allow any exit connections, make sure name resolution works (that is, your computer can resolve Internet addresses correctly). If there are any resources that your computer can't reach (for example, you are behind a restrictive firewall or content filter), please explicitly reject them in your exit policy — otherwise Tor users will be impacted too.

Why doesn't my Windows (or other OS) Tor relay run well?

Tor relays work best on Linux, FreeBSD 5.x+, OS X Tiger or later, and Windows Server 2003 or later.

You can probably get it working just fine on other operating systems too, but note the following caveats:

- Versions of Windows without the word "server" in their name sometimes have problems. This is especially the case for Win98, but it also happens in some cases for XP, especially if you don't have much memory. The problem is that we don't use the networking system calls in a very Windows-like way, so we run out of space in a fixed-size memory space known as the non-page pool, and then everything goes bad. The symptom is an assert error with the message "No buffer space available [WSAENOBUFS] [10055]". You can read more here.
- Most developers who contribute to Tor work with Unix-like operating systems. It would be great if more people with Windows experience help out, so we can improve Tor's usability and stability in Windows.
- More esoteric or archaic operating systems, like SunOS 5.9 or Irix64, may have problems with some libevent methods (devpoll, etc), probably due to bugs in libevent. If you experience crashes, try setting the EVENT_NODEVPOLL or equivalent environment variable.

Should I install Tor from my package manager, or build from source?

If you're using Debian or Ubuntu especially, there are a number of benefits to installing Tor from the Tor Project's repository.

- Your ulimit -n gets set to 32768 — high enough for Tor to keep open all the connections it needs.
- A user profile is created just for Tor, so Tor doesn't need to run as root.
- An init script is included so that Tor runs at boot.
- Tor runs with --verify-config, so that most problems with your config file get caught.
- Tor can bind to low level ports, then drop privileges.

What is the BadExit flag?

When an exit is misconfigured or malicious it's assigned the BadExit flag. This tells Tor to avoid exiting through that relay. In effect, relays with this flag become non-exits.

I got the BadExit flag why did that happen?

If you got this flag then we either discovered a problem or suspicious activity coming from your exit and weren't able to contact you. The reason for most flaggings are documented on the bad relays wiki. Please contact us so we can sort out the issue.

My relay recently got the Guard flag and traffic dropped by half.

Since it's now a guard, clients are using it less in other positions, but not many clients have rotated their existing guards out to use it as a guard yet. Read more details in this blog post or in Changing of the Guards: A Framework for Understanding and Improving Entry Guard Selection in Tor.

I want to run my Tor client on a different computer than my applications.

By default, your Tor client only listens for applications that connect from localhost. Connections from other computers are refused. If you want to torify applications on different computers than the Tor client, you should edit your torrc to define SocksListenAddress 0.0.0.0 and then restart (or hup) Tor. If you want to get more advanced, you can configure your Tor client on a firewall to bind to your internal IP but not your external IP.

Can I install Tor on a central server, and have my clients connect to it?

Yes. Tor can be configured as a client or a relay on another machine, and allow other machines to be able to connect to it for anonymity. This is most useful in an environment where many computers want a gateway of anonymity to the rest of the world. However, be forwarned that with this configuration, anyone within your private network (existing between you and the Tor client/relay) can see what traffic you are sending in clear text. The anonymity doesn't start until you get to the Tor relay. Because of this, if you are the controller of your domain and you know everything's locked down, you will be OK, but this configuration may not be suitable for large private networks where security is key all around.

Configuration is simple, editing your torrc file's SocksListenAddress according to the following examples:

```
SocksListenAddress 127.0.0.1

SocksListenAddress 192.168.x.x:9100

SocksListenAddress 0.0.0.0:9100
```

You can state multiple listen addresses, in the case that you are part of several networks or subnets.

```
SocksListenAddress 192.168.x.x:9100 #eth0
SocksListenAddress 10.x.x.x:9100 #eth1
```

After this, your clients on their respective networks/subnets would specify a socks proxy with the address and port you specified SocksListenAddress to be.

Please note that the SocksPort configuration option gives the port ONLY for localhost (127.0.0.1). When setting up your SocksListenAddress(es), you need to give the port with the address, as shown above.

If you are interested in forcing all outgoing data through the central Tor client/relay, instead of the server only being an optional proxy, you may find the program iptables (for *nix) useful.

Should I be a normal relay or bridge relay?

Bridge relays (or "bridges" for short) are Tor relays that aren't listed in the public Tor directory. That means that ISPs or governments trying to block access to the Tor network can't simply block all bridges.

Being a normal relay vs being a bridge relay is almost the same configuration: it's just a matter of whether your relay is listed publicly or not.

So bridges are useful a) for Tor users in oppressive regimes, and b) for people who want an extra layer of security because they're worried somebody will recognize that it's a public Tor relay IP address they're contacting.

Several countries, including China and Iran, have found ways to detect and block connections to Tor bridges. Obfsproxy bridges address this by adding another layer of obfuscation.

So should you run a normal relay or bridge relay? If you have lots of bandwidth, you should definitely run a normal relay. If you're willing to be an exit, you should definitely run a normal relay, since we need more exits. If you can't be an exit and only have a little bit of bandwidth, be a bridge. Thanks for volunteering!

I want to upgrade/move my relay. How do I keep the same key?

When upgrading your Tor relay, or running it on a different computer, the important part is to keep the same identity key (stored in "keys/secret_id_key" in your DataDirectory).

This means that if you're upgrading your Tor relay and you keep the same torrc and the same DataDirectory, then the upgrade should just work and your relay will keep using the same key. If you need to pick a new DataDirectory, be sure to copy your old keys/secret_id_key over.

How do I run my Tor relay as an NT service?

You can run Tor as a service on all versions of Windows except Windows 95/98/ME.

If you've already configured your Tor to be a relay, please note that when you enable Tor as a service, it will use a different DatagDirectory, and thus will generate a different key. If you want to keep using the old key, see the Upgrading your Tor relay FAQ entry for how to restore the old identity key.

To install Tor as a service, you can simply run:

```
tor --service install
```

A service called Tor Win32 Service will be installed and started. This service will also automatically start every time Windows boots, unless you change the Start-up type. An easy way to check the status of Tor, start or stop the service, and change the start-up type is by running services.msc and finding the Tor service in the list of currently installed services.

Optionally, you can specify additional options for the Tor service using the -options argument. For example, if you want Tor to use C:\tor\torrc, instead of the default torrc, and open a control port on port 9151, you would run:

```
tor --service install -options -f C:\tor\torrc ControlPort 9151
```

You can also start or stop the Tor service from the command line by typing:

```
 tor --service start
```

or

```
 tor --service stop
```

To remove the Tor service, you can run the following command:

```
tor --service remove
```

If you are running Tor as a service and you want to uninstall Tor entirely, be sure to run the service removal command (shown above) first before running the uninstaller from "Add/Remove Programs". The uninstaller is currently not capable of removing the active service.

Can I run a Tor relay from my virtual server account?

Some ISPs are selling "vserver" accounts that provide what they call a virtual server -- you can't actually interact with the hardware, and they can artificially limit certain resources such as the number of file descriptors you can open at once. Competent vserver admins are able to configure your server to not hit these limits. For example, in SWSoft's Virtuozzo, investigate /proc/user_beancounters. Look for "failcnt" in tcpsndbuf, tcprecvbuf, numothersock, and othersockbuf. Ask for these to be increased accordingly. Xen, Virtual Box and VMware virtual servers have no such limits normally.

If the vserver admin will not increase system limits another option is to reduce the memory allocated to the send and receive buffers on TCP connections Tor uses. An experimental feature to constrain socket buffers has recently been added. If your version of Tor supports it, set "ConstrainedSockets 1" in your configuration. See the tor man page for additional details about this option.

Unfortunately, since Tor currently requires you to be able to connect to all the other Tor relays, we need you to be able to use at least 1024 file descriptors. This means we can't make use of Tor relays that are crippled in this way.

We hope to fix this in the future, once we know how to build a Tor network with restricted topologies -- that is, where each node connects to only a few other nodes. But this is still a long way off.

I want to run more than one relay.

Great. If you want to run several relays to donate more to the network, we're happy with that. But please don't run more than a few dozen on the same network, since part of the goal of the Tor network is dispersal and diversity.

If you do decide to run more than one relay, please set the "MyFamily" config option in the torrc of each relay, listing all the relays (comma-separated) that are under your control:

```
    MyFamily $fingerprint1,$fingerprint2,$fingerprint3
```

where each fingerprint is the 40 character identity fingerprint (without spaces).

That way clients will know to avoid using more than one of your relays in a single circuit. You should set MyFamily if you have administrative control of the computers or of their network, even if they're not all in the same geographic location.

My relay is picking the wrong IP address.

Tor guesses its IP address by asking the computer for its hostname, and then resolving that hostname. Often people have old entries in their /etc/hosts file that point to old IP addresses.

If that doesn't fix it, you should use the "Address" config option to specify the IP you want it to pick. If your computer is behind a NAT and it only has an internal IP address, see the following FAQ entry on dynamic IP addresses.

Also, if you have many addresses, you might also want to set "OutboundBindAddress" so external connections come from the IP you intend to present to the world.

I'm behind a NAT/Firewall.

See http://portforward.com/ for directions on how to port forward with your NAT/router device.

If your relay is running on a internal net you need to setup port forwarding. Forwarding TCP connections is system dependent but the firewalled-clients FAQ entry offers some examples on how to do this.

Also, here's an example of how you would do this on GNU/Linux if you're using iptables:

```
/sbin/iptables -A INPUT -i eth0 -p tcp --destination-port 9001 -j ACCEPT
```

You may have to change "eth0" if you have a different external interface (the one connected to the Internet). Chances are you have only one (except the loopback) so it shouldn't be too hard to figure out.

Why is my Tor relay using so much memory?

If your Tor relay is using more memory than you'd like, here are some tips for reducing its footprint:

a. If you're on Linux, you may be encountering memory fragmentation bugs in glibc's malloc implementation. That is, when Tor releases memory back to the system, the pieces of memory are fragmented so they're hard to reuse. The Tor tarball ships with OpenBSD's malloc implementation, which doesn't have as many fragmentation bugs (but the tradeoff is higher CPU load). You can tell Tor to use this malloc implementation instead: `./configure --enable-openbsd-malloc`
b. If you're running a fast relay, meaning you have many TLS connections open, you are probably losing a lot of memory to OpenSSL's internal buffers (38KB+ per socket). We've patched OpenSSL to release unused buffer memory more aggressively. If you update to OpenSSL 1.0.0 or newer, Tor's build process will automatically recognize and use this feature.
c. If you still can't handle the memory load, consider reducing the amount of bandwidth your relay advertises. Advertising less bandwidth means you will attract fewer users, so your relay shouldn't grow as large. See the `MaxAdvertisedBandwidth` option in the man page.

All of this said, fast Tor relays do use a lot of ram. It is not unusual for a fast exit relay to use 500-1000 MB of memory.

Do I get better anonymity if I run a relay?

Yes, you do get better anonymity against some attacks.

The simplest example is an attacker who owns a small number of Tor relays. He will see a connection from you, but he won't be able to know whether the connection originated at your computer or was relayed from somebody else.

There are some cases where it doesn't seem to help: if an attacker can watch all of your incoming and outgoing traffic, then it's easy for him to learn which connections were relayed and which started at you. (In this case he still doesn't know your destinations unless he is watching them too, but you're no better off than if you were an ordinary client.)

There are also some downsides to running a Tor relay. First, while we only have a few hundred relays, the fact that you're running one might signal to an attacker that you place a high value on your anonymity. Second, there are some more esoteric attacks that are not as well-understood or well-tested that involve making use of the knowledge that you're running a relay -- for example, an attacker may be able to "observe" whether you're sending traffic even if he can't actually watch your network, by relaying traffic through your Tor relay and noticing changes in traffic timing.

It is an open research question whether the benefits outweigh the risks. A lot of that depends on the attacks you are most worried about. For most users, we think it's a smart move.

I'm facing legal trouble. How do I prove that my server was a Tor relay at a given time?

Exonerator is a web service that can check if an IP address was a relay at a given time. We can also provide a signed letter if needed.

Can I donate for a relay rather than run my own?

Sure! We recommend these non-profit charities that are happy to turn your donations into better speed and anonymity for the Tor network:

- torservers.net is a German charitable non-profit that runs a wide variety of exit relays worldwide. They also like donations of bandwidth from ISPs.
- Noisebridge is a US-based 501(c)(3) non-profit that collects donations and turns them into more US-based exit relay capacity.
- Nos Oignons is a French charitable non-profit that runs fast exit relays in France.
- DFRI is a Swedish non-profit running exit relays.

These organizations are not the same as The Tor Project, Inc, but we consider that a good thing. They're run by nice people who are part of the Tor community.

Note that there can be a tradeoff here between anonymity and performance. The Tor network's anonymity comes in part from diversity, so if you are in a position to run your own relay, you will be improving Tor's anonymity more than by donating. At the same time though, economies of scale for bandwidth mean that combining many small donations into several larger relays is more efficient at improving network performance. Improving anonymity and improving performance are both worthwhile goals, so however you can help is great!

Tor hidden services:

How do I access hidden services?

Tor hidden services are named with a special top-level domain (TLD) name in DNS: .onion. Since the .onion TLD is not recognized by the official root DNS servers on the Internet, your application will not get the response it needs to locate the service. Currently, the Tor directory server provides this look-up service; and thus the look-up request must get to the Tor network.

Therefore, your application **needs** to pass the .onion hostname to Tor directly. You can't try to resolve it to an IP address, since there *is* no corresponding IP address: the server is hidden, after all!

So, how do you make your application pass the hostname directly to Tor? You can't use SOCKS 4, since SOCKS 4 proxies require an IP from the client (a web browser is an example of a SOCKS client). Even though SOCKS 5 can accept either an IP or a hostname, most applications supporting SOCKS 5 try to resolve the name before passing it to the SOCKS proxy. SOCKS 4a, however, always accepts a hostname: You'll need to use SOCKS 4a.

Some applications, such as the browsers Mozilla Firefox and Apple's Safari, support sending DNS queries to Tor's SOCKS 5 proxy. Most web browsers don't support SOCKS 4a very well, though. The workaround is to point your web browser at an HTTP proxy, and tell the HTTP proxy to speak to Tor with SOCKS 4a. We recommend Polipo as your HTTP proxy.

For applications that do not support HTTP proxy, and so cannot use Polipo, FreeCap is an alternative. When using FreeCap set proxy protocol to SOCKS 5 and under settings set DNS name resolving to remote. This will allow you to use almost any program with Tor without leaking DNS lookups and allow those same programs to access hidden services.

See also the question on DNS.

How do I provide a hidden service?

See the official hidden service configuration instructions.

Development:

What do these weird version numbers mean?

Versions of Tor before 0.1.0 used a strange and hard-to-explain version scheme. Let's forget about those.

Starting with 0.1.0, versions all look like this: MAJOR.MINOR.MICRO(.PATCHLEVEL)(-TAG). The stuff in parenthesis is optional. MAJOR, MINOR, MICRO, and PATCHLEVEL are all numbers. Only one release is ever made with any given set of these version numbers. The TAG lets you know how stable we think the release is: "alpha" is pretty unstable; "rc" is a release candidate; and no tag at all means that we have a final release. If the tag ends with "-cvs", you're looking at a development snapshot that came after a given release.

So for example, we might start a development branch with (say) 0.1.1.1-alpha. The patchlevel increments consistently as the status tag changes, for example, as in: 0.1.1.2-alpha, 0.1.1.3-alpha, 0.1.1.4-rc, 0.1.1.5-rc, etc. Eventually, we would release 0.1.1.6. The next stable release would be 0.1.1.7.

Why do we do it like this? Because every release has a unique version number, it is easy for tools like package manager to tell which release is newer than another. The tag makes it easy for users to tell how stable the release is likely to be.

How do I set up my own private Tor network?

If you want to experiment locally with your own network, or you're cut off from the Internet and want to be able to mess with Tor still, then you may want to set up your own separate Tor network.

To set up your own Tor network, you need to run your own authoritative directory servers, and your clients and relays must be configured so they know about your directory servers rather than the default public ones.

Apart from the somewhat tedious method of manually configuring a couple of directory authorities, relays and clients there are two separate tools that could help. One is Chutney, the other is Shadow.

Chutney is a tool for configuring, controlling and running tests on a testing Tor network. It requires that you have Tor and Python (2.5 or later) installed on your system. You can use Chutney to create a testing network by generating Tor configuration files (torrc) and necssary keys (for the directory authorities). Then you can let Chutney start your Tor authorities, relays and clients and wait for the network to bootstrap. Finally, you can have Chutney run tests on your network to see which things work and which do not. Chutney is typically used for running a testing network with about 10 instances of Tor. Every instance of Tor binds to one or two ports on localhost (127.0.0.1) and all Tor communication is done over the loopback interface. The Chutney README is a good starting point for getting it up and running.

Shadow is a network simulator that can run Tor through its Scallion plug-in. Although it's typically used for running load and performance tests on substantially larger Tor test networks than what's feasible with Chutney, it also makes for an excellent debugging tool since you can run completely deterministic experiments. A large Shadow network is on the size of thousands of instances of Tor, and you can run experiments out of the box using one of Shadow's several included scallion experiment configurations. Shadow can be run on any linux machine without root, and can also run on EC2 using a pre-configured image. Also, Shadow controls the time of the simulation with the effect that time-consuming tests can be done more efficiently than in an ordinary testing network. The Shadow wiki and Shadow website are good places to get started.

How can I make my Java program use the Tor Network?

The newest versions of Java now have SOCKS4/5 support built in. Unfortunately, the SOCKS interface is not very well documented and may still leak your DNS lookups. The safest way to use Tor is to interface the SOCKS protocol directly or go through an application-level proxy that speaks SOCKS4a. For an example and libraries that implement the SOCKS4a connection, go to Joe Foley's TorLib in the TinFoil Project.

A fully Java implementation of the Tor client is now available as Orchid. We still consider Orchid to be experimental, so use with care.

What is Libevent?

When you want to deal with a bunch of net connections at once, you have a few options:

One is multithreading: you have a separate micro-program inside the main program for each net connection that reads and writes to the connection as needed.This, performance-wise, sucks.

Another is asynchronous network programming: you have a single main program that finds out when various net connections are ready to read/write, and acts accordingly.

The problem is that the oldest ways to find out when net connections are ready to read/write, suck. And the newest ways are finally fast, but are not available on all platforms.

This is where Libevent comes in and wraps all these ways to find out whether net connections are ready to read/write, so that Tor (and other programs) can use the fastest one that your platform supports, but can still work on older platforms (these methods are all different depending on the platorm) So Libevent presents a consistent and fast interface to select, poll, kqueue, epoll, /dev/poll, and windows select.

However, On the the Win32 platform (by Microsoft) the only good way to do fast IO on windows with hundreds of sockets is using overlapped IO, which is grossly unlike every other BSD sockets interface.

Libevent has its own website.

What do I need to do to get a new feature into Tor?

For a new feature to go into Tor, it needs to be designed (explain what you think Tor should do), argued to be secure (explain why it's better or at least as good as what Tor does now), specified (explained at the byte level at approximately the level of detail in tor-spec.txt), and implemented (done in software).

You probably shouldn't count on other people doing all of these steps for you: people who are skilled enough to do this stuff generally have their own favorite feature requests.

Anonymity And Security:

What protections does Tor provide?

Internet communication is based on a store-and-forward model that can be understood in analogy to postal mail: Data is transmitted in blocks called IP datagrams or packets. Every packet includes a source IP address (of the sender) and a destination IP address (of the receiver), just as ordinary letters contain postal addresses of sender and receiver. The way from sender to receiver involves multiple hops of routers, where each router inspects the destination IP address and forwards the packet closer to its destination. Thus, every router between sender and receiver learns that the sender is communicating with the receiver. In particular, your local ISP is in the position to build a complete profile of your Internet usage. In addition, every server in the Internet that can see any of the packets can profile your behaviour.

The aim of Tor is to improve your privacy by sending your traffic through a series of proxies. Your communication is encrypted in multiple layers and routed via multiple hops through the Tor network to the final receiver. More details on this process can be found in the Tor overview. Note that all your local ISP can observe now is that you are communicating with Tor nodes. Similarly, servers in the Internet just see that they are being contacted by Tor nodes.

Generally speaking, Tor aims to solve three privacy problems:

First, Tor prevents websites and other services from learning your location, which they can use to build databases about your habits and interests. With Tor, your Internet connections don't give you away by default -- now you can have the ability to choose, for each connection, how much information to reveal.

Second, Tor prevents people watching your traffic locally (such as your ISP) from learning what information you're fetching and where you're fetching it from. It also stops them from deciding what you're allowed to learn and publish -- if you can get to any part of the Tor network, you can reach any site on the Internet.

Third, Tor routes your connection through more than one Tor relay so no single relay can learn what you're up to. Because these relays are run by different individuals or organizations, distributing trust provides more security than the old one hop proxy approach.

Note, however, that there are situations where Tor fails to solve these privacy problems entirely: see the entry below on remaining attacks.

Can exit nodes eavesdrop on communications? Isn't that bad?

Yes, the guy running the exit node can read the bytes that come in and out there. Tor anonymizes the origin of your traffic, and it makes sure to encrypt everything inside the Tor network, but it does not magically encrypt all traffic throughout the Internet.

This is why you should always use end-to-end encryption such as SSL for sensitive Internet connections. (The corollary to this answer is that if you are worried about somebody intercepting your traffic and you're *not* using end-to-end encryption at the application layer, then something has already gone wrong and you shouldn't be thinking that Tor is the problem.)

Tor does provide a partial solution in a very specific situation, though. When you make a connection to a destination that also runs a Tor relay, Tor will automatically extend your circuit so you exit from that circuit. So for example if Indymedia ran a Tor relay on the same IP address as their website, people using Tor to get to the Indymedia website would automatically exit from their Tor relay, thus getting *better* encryption and authentication properties than just browsing there the normal way.

We'd like to make it still work even if the service is nearby the Tor relay but not on the same IP address. But there are a variety of technical problems we need to overcome first (the main one being "how does the Tor client learn which relays are associated with which websites in a decentralized yet non-gamable way?").

So I'm totally anonymous if I use Tor?

No.

First, Tor protects the network communications. It separates where you are from where you are going on the Internet. What content and data you transmit over Tor is controlled by you. If you login to Google or Facebook via Tor, the local ISP or network provider doesn't know you are visiting Google or Facebook. Google and Facebook don't know where you are in the world. However, since you have logged into their sites, they know who you are. If you don't want to share information, you are in control.

Second, active content, such as Java, Javascript, Adobe Flash, Adobe Shockwave, QuickTime, RealAudio, ActiveX controls, and VBScript, are binary applications. These binary applications run as your user account with your permissions in your operating system. This means these applications can access anything that your user account can access. Some of these technologies, such as Java and Adobe Flash for instance, run in what is known as a virtual machine. This virtual machine may have the ability to ignore your configured proxy settings, and therefore bypass Tor and share information directly to other sites on the Internet. The virtual machine may be able to store data, such as cookies, completely separate from your browser or operating system data stores. Therefore, these technologies must be disabled in your browser to use Tor safely.

That's where Tor Browser comes in. We produce a web browser that is preconfigured to help you control the risks to your privacy and anonymity while browsing the Internet. Not only are the above technologies disabled to prevent identity leaks, the Tor Browser also includes browser extensions like NoScript and Torbutton, as well as patches to the Firefox source code. The full design of the Tor Browser can be read here. In designing a safe, secure solution for browsing the web with Tor, we've discovered that configuring other browsers to use Tor is unsafe.

Alternatively, you may find a Live CD or USB operating system more to your liking. The Tails team has created an entire bootable operating system configured for anonymity and privacy on the Internet.

Tor is a work in progress. There is still plenty of work left to do for a strong, secure, and complete solution.

Tell me about all the keys Tor uses.

Tor uses a variety of different keys, with three goals in mind: 1) encryption to ensure privacy of data within the Tor network, 2) authentication so clients know they're talking to the relays they meant to talk to, and 3) signatures to make sure all clients know the same set of relays.

Encryption: first, all connections in Tor use TLS link encryption, so observers can't look inside to see which circuit a given cell is intended for. Further, the Tor client establishes an ephemeral encryption key with each relay in the circuit; these extra layers of encryption mean that only the exit relay can read the cells. Both sides discard the circuit key when the circuit ends, so logging traffic and then breaking into the relay to discover the key won't work.

Authentication: Every Tor relay has a public decryption key called the "onion key". Each relay rotates its onion key once a week. When the Tor client establishes circuits, at each step it demands that the Tor relay prove knowledge of its onion key. That way the first node in the path can't just spoof the rest of the path. Because the Tor client chooses the path, it can make sure to get Tor's "distributed trust" property: no single relay in the path can know about both the client and what the client is doing.

Coordination: How do clients know what the relays are, and how do they know that they have the right keys for them? Each relay has a long-term public signing key called the "identity key". Each directory authority additionally has a "directory signing key". The directory authorities provide a signed list of all the known relays, and in that list are a set of certificates from each relay (self-signed by their identity key) specifying their keys, locations, exit policies, and so on. So unless the adversary can control a majority of the directory authorities (as of 2012 there are 8 directory authorities), he can't trick the Tor client into using other Tor relays.

How do clients know what the directory authorities are? The Tor software comes with a built-in list of location and public key for each directory authority. So the only way to trick users into using a fake Tor network is to give them a specially modified version of the software.

How do users know they've got the right software? When we distribute the source code or a package, we digitally sign it with GNU Privacy Guard. See the instructions on how to check Tor's signatures.

In order to be certain that it's really signed by us, you need to have met us in person and gotten a copy of our GPG key fingerprint, or you need to know somebody who has. If you're concerned about an attack on this level, we recommend you get involved with the security community and start meeting people.

What are Entry Guards?

Tor (like all current practical low-latency anonymity designs) fails when the attacker can see both ends of the communications channel. For example, suppose the attacker controls or watches the Tor relay you choose to enter the network, and also controls or watches the website you visit. In this case, the research community knows no practical low-latency design that can reliably stop the attacker from correlating volume and timing information on the two sides.

So, what should we do? Suppose the attacker controls, or can observe, *C* relays. Suppose there are *N* relays total. If you select new entry and exit relays each time you use the network, the attacker will be able to correlate all traffic you send with probability around *(c/n)2.* But profiling is, for most users, as bad as being traced all the time: they want to do something often without an attacker noticing, and the attacker noticing once is as bad as the attacker noticing more often. Thus, choosing many random entries and exits gives the user no chance of escaping profiling by this kind of attacker.

The solution is "entry guards": each Tor client selects a few relays at random to use as entry points, and uses only those relays for her first hop. If those relays are not controlled or observed, the attacker can't win, ever, and the user is secure. If those relays *are* observed or controlled by the attacker, the attacker sees a larger *fraction* of the user's traffic — but still the user is no more profiled than before. Thus, the user has some chance (on the order of *(n-c)/n*) of avoiding profiling, whereas she had none before.

You can read more at An Analysis of the Degradation of Anonymous Protocols, Defending Anonymous Communication Against Passive Logging Attacks, and especially Locating Hidden Servers.

Restricting your entry nodes may also help against attackers who want to run a few Tor nodes and easily enumerate all of the Tor user IP addresses. (Even though they can't learn what destinations the users are talking to, they still might be able to do bad things with just a list of users.) However, that feature won't really become useful until we move to a "directory guard" design as well.

How often does Tor change its paths?

Tor will reuse the same circuit for new TCP streams for 10 minutes, as long as the circuit is working fine. (If the circuit fails, Tor will switch to a new circuit immediately.)

But note that a single TCP stream (e.g. a long IRC connection) will stay on the same circuit forever -- we don't rotate individual streams from one circuit to the next. Otherwise an adversary with a partial view of the network would be given many chances over time to link you to your destination, rather than just one chance.

Tor uses hundreds of bytes for every IRC line. I can't afford that!

Tor sends data in chunks of 512 bytes (called "cells"), to make it harder for intermediaries to guess exactly how many bytes you're communicating at each step. This is unlikely to change in the near future -- if this increased bandwidth use is prohibitive for you, I'm afraid Tor is not useful for you right now.

The actual content of these fixed size cells is documented in the main Tor spec, section 3.

We have been considering one day adding two classes of cells -- maybe a 64 byte cell and a 1024 byte cell. This would allow less overhead for interactive streams while still allowing good throughput for bulk streams. But since we want to do a lot of work on quality-of-service and better queuing approaches first, you shouldn't expect this change anytime soon (if ever). However if you are keen, there are a couple of research ideas that may involve changing the cell size.

Why does netstat show these outbound connections?

Because that's how Tor works. It holds open a handful of connections so there will be one available when you need one.

What about powerful blocking mechanisms?

An adversary with a great deal of manpower and money, and severe real-world penalties to discourage people from trying to evade detection, is a difficult test for an anonymity and anti-censorship system.

The original Tor design was easy to block if the attacker controls Alice's connection to the Tor network --- by blocking the directory authorities, by blocking all the relay IP addresses in the directory, or by filtering based on the fingerprint of the Tor TLS handshake. After seeing these attacks and others first-hand, more effort was put into researching new circumvention techniques. Pluggable transports are protocols designed to allow users behind government firewalls to access the Tor network.

We've made quite a bit of progress on this problem lately. You can read more details on the pluggable transports page. You may also be interested in Roger and Jake's talk at 28C3, or Runa's talk at 44con.

Does Tor resist "remote physical device fingerprinting"?

Yes, we resist all of these attacks as far as we know.

These attacks come from examining characteristics of the IP headers or TCP headers and looking for information leaks based on individual hardware signatures. One example is the Oakland 2005 paper that lets you learn if two packet streams originated from the same hardware, but only if you can see the original TCP timestamps.

Tor transports TCP streams, not IP packets, so we end up automatically scrubbing a lot of the potential information leaks. Because Tor relays use their own (new) IP and TCP headers at each hop, this information isn't relayed from hop to hop. Of course, this also means that we're limited in the protocols we can transport (only correctly-formed TCP, not all IP like ZKS's Freedom network could) -- but maybe that's a good thing at this stage.

Is Tor like a VPN?

Do not use a VPN as an anonymity solution. If you're looking for a trusted entry into the Tor network, or if you want to obscure the fact that you're using Tor, setting up a private server as a bridge works quite well.

VPNs encrypt the traffic between the user and the VPN provider, and they can act as a proxy between a user and an online destination. However, VPNs have a single point of failure: the VPN provider. A technically proficient attacker or a number of employees could retrieve the full identity information associated with a VPN user. It is also possible to use coercion or other means to convince a VPN provider to reveal their users' identities. Identities can be discovered by following a money trail (using Bitcoin does not solve this problem because Bitcoin is not anonymous), or by persuading the VPN provider to hand over logs. Even if a VPN provider says they don't keep logs, users have to take their word for it---and trust that the VPN provider won't buckle to outside pressures that might want them to start keeping logs.

When you use a VPN, websites can still build up a persistent profile of your usage over time. Even though sites you visit won't automatically get your originating IP address, they still know how to profile you based on your browsing history.

When you use Tor the IP address you connect to changes at most every 10 minutes, and often more frequently than that. This makes it extremely dificult for websites to create any sort of persistent profile of Tor users (assuming you did not identify yourself in other ways). No one Tor relay can know enough information to compromise any Tor user because of Tor's encrypted three-hop circuit design.

Aren't 10 proxies (proxychains) better than Tor with only 3 hops?

Proxychains is a program that sends your traffic through a series of open web proxies that you supply before sending it on to your final destination. Unlike Tor, proxychains does not encrypt the connections between each proxy server. An open proxy that wanted to monitor your connection could see all the other proxy servers you wanted to use between itself and your final destination, as well as the IP address that proxy hop received traffic from.

Because the Tor protocol requires encrypted relay-to-relay connections, not even a misbehaving relay can see the entire path of any Tor user.

While Tor relays are run by volunteers and checked periodically for suspicious behavior, many open proxies that can be found with a search engine are compromised machines, misconfigured private proxies not intended for public use, or honeypots set up to exploit users.

What attacks remain against onion routing?

As mentioned above, it is possible for an observer who can view both you and either the destination website or your Tor exit node to correlate timings of your traffic as it enters the Tor network and also as it exits. Tor does not defend against such a threat model.

In a more limited sense, note that if a censor or law enforcement agency has the ability to obtain specific observation of parts of the network, it is possible for them to verify a suspicion that you talk regularly to your friend by observing traffic at both ends and correlating the timing of only that traffic. Again, this is only useful to verify that parties already suspected of communicating with one another are doing so. In most countries, the suspicion required to obtain a warrant already carries more weight than timing correlation would provide.

Furthermore, since Tor reuses circuits for multiple TCP connections, it is possible to associate non anonymous and anonymous traffic at a given exit node, so be careful about what applications you run concurrently over Tor. Perhaps even run separate Tor clients for these applications.

Where can I learn more about anonymity?

Read these papers (especially the ones in boxes) to get up to speed on anonymous communication systems.

Alternate designs:

You should make every Tor user be a relay.

Requiring every Tor user to be a relay would help with scaling the network to handle all our users, and running a Tor relay may help your anonymity. However, many Tor users cannot be good relays — for example, some Tor clients operate from behind restrictive firewalls, connect via modem, or otherwise aren't in a position where they can relay traffic. Providing service to these clients is a critical part of providing effective anonymity for everyone, since many Tor users are subject to these or similar constraints and including these clients increases the size of the anonymity set.

That said, we do want to encourage Tor users to run relays, so what we really want to do is simplify the process of setting up and maintaining a relay. We've made a lot of progress with easy configuration in the past few years: Tor is good at automatically detecting whether it's reachable and how much bandwidth it can offer.

There are five steps we need to address before we can do this though:

First, we need to make Tor stable as a relay on all common operating systems. The main remaining platform is Windows, and we're mostly there. See Section 4.1 of our development roadmap.

Second, we still need to get better at automatically estimating the right amount of bandwidth to allow. See item #7 on the research section of the volunteer page: "Tor doesn't work very well when relays have asymmetric bandwidth (e.g. cable or DSL)". It might be that switching to UDP transport is the simplest answer here — which alas is not a very simple answer at all.

Third, we need to work on scalability, both of the network (how to stop requiring that all Tor relays be able to connect to all Tor relays) and of the directory (how to stop requiring that all Tor users know about all Tor relays). Changes like this can have large impact on potential and actual anonymity. See Section 5 of the Challenges paper for details. Again, UDP transport would help here.

Fourth, we need to better understand the risks from letting the attacker send traffic through your relay while you're also initiating your own anonymized traffic. Three different research papers describe ways to identify the relays in a circuit by running traffic through candidate relays and looking for dips in the traffic while the circuit is active. These clogging attacks are not that scary in the Tor context so long as relays are never clients too. But if we're trying to encourage more clients to turn on relay functionality too (whether as bridge relays or as normal relays), then we need to understand this threat better and learn how to mitigate it.

Fifth, we might need some sort of incentive scheme to encourage people to relay traffic for others, and/or to become exit nodes. Here are our current thoughts on Tor incentives.

Please help on all of these!

You should transport all IP packets, not just TCP packets.

This would be handy, because it would make Tor better able to handle new protocols like VoIP, it could solve the whole need to socksify applications, and it would solve the fact that exit relays need to allocate a lot of file descriptors to hold open all the exit connections.

We're heading in this direction: see this trac ticket for directions we should investigate. Some of the hard problems are:

a. IP packets reveal OS characteristics. We would still need to do IP-level packet normalization, to stop things like TCP fingerprinting attacks. Given the diversity and complexity of TCP stacks, along with device fingerprinting attacks, it looks like our best bet is shipping our own user-space TCP stack.
b. Application-level streams still need scrubbing. We will still need user-side applications like Torbutton. So it won't become just a matter of capturing packets and anonymizing them at the IP layer.
c. Certain protocols will still leak information. For example, we must rewrite DNS requests so they are delivered to an unlinkable DNS server rather than the DNS server at a user's ISP; thus, we must understand the protocols we are transporting.
d. DTLS (datagram TLS) basically has no users, and IPsec sure is big. Once we've picked a transport mechanism, we need to design a new end-to-end Tor protocol for avoiding tagging attacks and other potential anonymity and integrity issues now that we allow drops, resends, et cetera.
e. Exit policies for arbitrary IP packets mean building a secure IDS. Our node operators tell us that exit policies are one of the main reasons they're willing to run Tor. Adding an Intrusion Detection System to handle exit policies would increase the security complexity of Tor, and would likely not work anyway, as evidenced by the entire field of IDS and counter-IDS papers. Many potential abuse issues are resolved by the fact that Tor only transports valid TCP streams (as opposed to arbitrary IP including malformed packets and IP floods), so exit policies become even *more* important as we become able to transport IP packets. We also need to compactly describe exit policies in the Tor directory, so clients can predict which nodes will allow their packets to exit — and clients need to predict all the packets they will want to send in a session before picking their exit node!
f. The Tor-internal name spaces would need to be redesigned. We support hidden service ".onion" addresses by intercepting the addresses when they are passed to the Tor client. Doing so at the IP level will require a more complex interface between Tor and the local DNS resolver.

You should hide the list of Tor relays, so people can't block the exits.

There are a few reasons we don't:

a. We can't help but make the information available, since Tor clients need to use it to pick their paths. So if the "blockers" want it, they can get it anyway. Further, even if we didn't tell clients about the list of relays directly, somebody could still make a lot of connections through Tor to a test site and build a list of the addresses they see.
b. If people want to block us, we believe that they should be allowed to do so. Obviously, we would prefer for everybody to allow Tor users to connect to them, but people have the right to decide who their services should allow connections from, and if they want to block anonymous users, they can.
c. Being blockable also has tactical advantages: it may be a persuasive response to website maintainers who feel threatened by Tor. Giving them the option may inspire them to stop and think about whether they really want to eliminate private access to their system, and if not, what other options they might have. The time they might otherwise have spent blocking Tor, they may instead spend rethinking their overall approach to privacy and anonymity.

You should let people choose their path length.

Right now the path length is hard-coded at 3 plus the number of nodes in your path that are sensitive. That is, in normal cases it's 3, but for example if you're accessing a hidden service or a ".exit" address it could be 4.

We don't want to encourage people to use paths longer than this — it increases load on the network without (as far as we can tell) providing any more security. Remember that the best way to attack Tor is to attack the endpoints and ignore the middle of the path. Also, using paths longer than 3 could harm anonymity, first because it makes "denial of security" attacks easier, and second because it could act as an identifier if only a few people do it ("Oh, there's that person who changed her path length again").

And we don't want to encourage people to use paths of length 1 either. Currently there is no reason to suspect that investigating a single relay will yield user-destination pairs, but if many people are using only a single hop, we make it more likely that attackers will seize or break into relays in hopes of tracing users.

Now, there is a good argument for making the number of hops in a path unpredictable. For example, somebody who happens to control the last two hops in your path still doesn't know who you are, but they know for sure which entry node you used. Choosing path length from, say, a geometric distribution will turn this into a statistical attack, which seems to be an improvement. On the other hand, a longer path length is bad for usability, and without further protections it seems likely that an adversary can estimate your path length anyway. We're not sure of the right trade-offs here. Please write a research paper that tells us what to do.

You should split each connection over many paths.

We don't currently think this is a good idea. You see, the attacks we're worried about are at the endpoints: the adversary watches Alice (or the first hop in the path) and Bob (or the last hop in the path) and learns that they are communicating.

If we make the assumption that timing attacks work well on even a few packets end-to-end, then having *more* possible ways for the adversary to observe the connection seems to hurt anonymity, not help it.

Now, it's possible that we could make ourselves more resistant to end-to-end attacks with a little bit of padding and by making each circuit send and receive a fixed number of cells. This approach is more well-understood in the context of high-latency systems. See e.g. Message Splitting Against the Partial Adversary by Andrei Serjantov and Steven J. Murdoch.

But since we don't currently understand what network and padding parameters, if any, could provide increased end-to-end security, our current strategy is to minimize the number of places that the adversary could possibly see.

You should migrate application streams across circuits.

This would be great for two reasons. First, if a circuit breaks, we would be able to shift its active streams onto a new circuit, so they don't have to break. Second, it is conceivable that we could get increased security against certain attacks by migrating streams periodically, since leaving a stream on a given circuit for many hours might make it more vulnerable to certain adversaries.

There are two problems though. First, Tor would need a much more bulky protocol. Right now each end of the Tor circuit just sends the cells, and lets TCP provide the in-order guaranteed delivery. If we can move streams across circuits, though, we would need to add queues at each end of the circuit, add sequence numbers so we can send and receive acknowledgements for cells, and so forth. These changes would increase the complexity of the Tor protocol considerably. Which leads to the second problem: if the exit node goes away, there's nothing we can do to save the TCP connection. Circuits are typically three hops long, so in about a third of the cases we just lose.

Thus our current answer is that since we can only improve things by at best 2/3, it's not worth the added code and complexity. If somebody writes a protocol specification for it and it turns out to be pretty simple, we'd love to add it.

But there are still some approaches we can take to improve the reliability of streams. The main approach we have now is to specify that streams using certain application ports prefer circuits to be made up of stable nodes. These ports are specified in the "LongLivedPorts" torrc option, and they default to

```
21,22,706,1863,5050,5190,5222,5223,6667,6697,8300
```

The definition of "stable" is an open research question, since we can only guess future stability based on past performance. Right now we judge that a node is stable if it advertises that it has been up for more than a day. Down the road we plan to refine this so it takes into account the average stability of the other nodes in the Tor network.

You should let the network pick the path, not the client

No. You cannot trust the network to pick the path for relays could collude and route you through their colluding friends. This would give an adversary the ability to watch all of your traffic end to end.

Your default exit policy should block unallocated net blocks too.

No, it shouldn't. The default exit policy blocks certain private net blocks, like 10.0.0.0/8, because they might actively be in use by Tor relays and we don't want to cause any surprises by bridging to internal networks. Some overzealous firewall configs suggest that you also block all the parts of the Internet that IANA has not currently allocated. First, this turns into a problem for them when those addresses *are* allocated. Second, why should we default-reject something that might one day be useful?

Tor's default exit policy is chosen to be flexible and useful in the future: we allow everything except the specific addresses and ports that we anticipate will lead to problems.

Exit policies should be able to block websites, not just IP addresses.

It would be nice to let relay operators say things like "reject www.slashdot.org" in their exit policies, rather than requiring them to learn all the IP address space that could be covered by the site (and then also blocking other sites at those IP addresses).

There are two problems, though. First, users could still get around these blocks. For example, they could request the IP address rather than the hostname when they exit from the Tor network. This means operators would still need to learn all the IP addresses for the destinations in question.

The second problem is that it would allow remote attackers to censor arbitrary sites. For example, if a Tor operator blocks www1.slashdot.org, and then some attacker poisons the Tor relay's DNS or otherwise changes that hostname to resolve to the IP address for a major news site, then suddenly that Tor relay is blocking the news site.

You should change Tor to prevent users from posting certain content.

Tor only transports data, it does not inspect the contents of the connections which are sent over it. In general it's a very hard problem for a computer to determine what is objectionable content with good true positive/false positive rates and we are not interested in addressing this problem.

Further, and more importantly, which definition of "certain content" could we use? Every choice would lead to a quagmire of conflicting personal morals. The only solution is to have no opinion.

You should send padding so it's more secure.

Like all anonymous communication networks that are fast enough for web browsing, Tor is vulnerable to statistical "traffic confirmation" attacks, where the adversary watches traffic at both ends of a circuit and confirms his guess that they're communicating. It would be really nice if we could use cover traffic to confuse this attack. But there are three problems here:

- Cover traffic is really expensive. And *every* user needs to be doing it. This adds up to a lot of extra bandwidth cost for our volunteer operators, and they're already pushed to the limit.
- You'd need to always be sending traffic, meaning you'd need to always be online. Otherwise, you'd need to be sending end-to-end cover traffic -- not just to the first hop, but all the way to your final destination -- to prevent the adversary from correlating presence of traffic at the destination to times when you're online. What does it mean to send cover traffic to -- and from -- a web server? That is not supported in most protocols.
- Even if you *could* send full end-to-end padding between all users and all destinations all the time, you're *still* vulnerable to active attacks that block the padding for a short time at one end and look for patterns later in the path.

In short, for a system like Tor that aims to be fast, we don't see any use for padding, and it would definitely be a serious usability problem. We hope that one day somebody will prove us wrong, but we are not optimistic.

You should use steganography to hide Tor traffic.

Many people suggest that we should use steganography to make it hard to notice Tor connections on the Internet. There are a few problems with this idea though:

First, in the current network topology, the Tor relays list is public and can be accessed by attackers. An attacker who wants to detect or block anonymous users could always just notice **any connection** to or from a Tor relay's IP address.

Abuse:

Doesn't Tor enable criminals to do bad things?

For the answer to this question and others, please see our Tor Abuse FAQ.

How do I respond to my ISP about my exit relay?

A collection of templates for successfully responding to ISPs is collected here.

I have questions about a Tor IP address for a legal case.

Please read the legal FAQ written by EFF lawyers. There's a growing legal directory of people who may be able to help you.

If you need to check if a certain IP address was acting as a Tor exit node at a certain date and time, you can use the ExoneraTor tool to query the historic Tor relay lists and get an answer.

Trademark, copyright notices, and rules for use by third parties can be found in our FAQ.

About Tor
- What Tor Does
- Users of Tor
- Core Tor People
- Sponsors
- Contact Us

Get Involved
- Donate
- Mailing Lists
- Mirrors
- Hidden Services
- Translations

Documentation
- Manuals
- Installation
- Guides
- Tor Wiki
- General Tor FAQ

Proud member of
THE INTERNET DEFENSE LEAGUE
EST 2012